LLM-Based Multi-Agent Systems are Scalable Graph Generative Models

Jiarui Ji[1], **Runlin Lei**[1], **Jialing Bi**[1],
Zhewei Wei[1 *], **Xu Chen**[1], **Yankai Lin**[1],
Xuchen Pan[2], **Yaliang Li**[2], **Bolin Ding**[2]
[1]Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
[2]Alibaba Group

Abstract

The structural properties of naturally arising social graphs are extensively studied to understand their evolution. Prior approaches for modeling network dynamics typically rely on rule-based models, which lack realism and generalizability, or deep learning-based models, which require large-scale training datasets. As abstract graph representations of entity-wise interactions, social graphs present an opportunity to explore network evolution mechanisms through realistic simulations of human-item interactions. Leveraging the pre-trained social consensus knowledge embedded in large language models (LLMs), we present GraphAgent-Generator (GAG), a novel simulation-based framework for dynamic, text-attributed social graph generation. GAG simulates the temporal node and edge generation processes for zero-shot social graph generation. The resulting graphs adhere to seven key macroscopic network properties, achieving an 11% improvement in microscopic graph structure metrics. Through the node classification benchmarking task, we validate that GAG effectively captures the intricate text-structure correlations in graph generation. Furthermore, GAG supports generating graphs with up to nearly 100,000 nodes or 10 million edges through large-scale LLM-based agent simulation with parallel acceleration, achieving a minimum speed-up of 90.4%. The source code is available at `https://github.com/Ji-Cather/GraphAgent`.

1 Introduction

Social graphs are mathematical structures stem from pairwise interactions between entities through nodes and edges, serving as a fundamental concept in network science. They are widely used to model human behaviors across various domains, including scientific research (Radicchi et al., 2011; Han et al., 2024), e-commerce (Zheng et al., 2025), and sociology (Yuan et al., 2024). A longstanding task in network science is social graph generation. Given an observed graph dataset, researchers construct models based on proposed generative mechanisms to extrapolate these observations into the growth of complex networks. Network science theories constitute macroscopic properties such as power-law degree distribution (Clauset et al., 2009). In contrast, microscopic properties include graph structure metrics like degree distribution and clustering coefficient (Martinkus et al., 2022). By comparing the macroscopic and microscopic properties of the generated and real-world graphs, researchers gain deeper insights into the mechanisms underlying graph evolution.

Existing graph generation methods can be categorized into two types: (1) Rule-based methods, which rely on preset rules to generate graphs (Erdos et al., 1960; Barabási and Albert, 1999). These methods are designed to capture specific macroscopic properties observed in real-world networks. However, the need for tailored models to capture each property complicates the integration of these methods into a unified framework. (Bergmeister et al., 2024) (2) Deep learning-based methods, which leverage self-supervised learning to capture graph structures. These methods mainly include sequential methods (You et al., 2018) and one-shot methods (Vignac et al., 2023; Bergmeister et al., 2024; Simonovsky and Komodakis, 2018). While these techniques excel in fitting the microscopic properties of observed graphs, they face challenges when generating larger graphs beyond the size of the observed dataset (Bergmeister et al., 2024) and struggle to maintain macroscopic properties during the growth of complex networks.

Since social graphs evolve over time, an ideal

* Zhewei Wei is the corresponding author.
The work was partially done at Beijing Key Laboratory of Research on Large Models and Intelligent Governance, MOE Key Lab of Data Engineering and Knowledge Engineering, Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE, and Pazhou Laboratory (Huangpu), Guangzhou, Guangdong 510555, China.

Findings of the Association for Computational Linguistics: ACL 2025, pages 1492–1523
July 27 - August 1, 2025

graph generator should understand the underlying physical processes that generalize to unseen nodes during graph expansion (Gupta et al., 2022) instead of simply following preset rules or fitting the training data. For social graphs specifically, the dynamics of human-item interactions drive the network evolution (Fowler and Christakis, 2010). Fortunately, the emergence of LLMs like LLaMA (AI@Meta, 2024) and GPT-4 (OpenAI, 2023) has opened new avenues for graph generation. With advanced capabilities in human-like decision-making, LLM-based agents can effectively simulate complex interaction processes in human activities (Park et al., 2023).

In this work, we introduce GraphAgent-Generator (GAG), a novel framework for social graph generation. Our approach draws on empirically studied concepts from the social sciences, particularly bipartite models of social graphs that capture actor-item interactions. In GAG, the actor set consists of LLM-based agents, while the initial item set is derived from a real-world seed graph. This item set is subsequently expanded through items generated by the actors. We propose the S-RAG algorithm to model actor-item interactions and simulate network growth patterns originating from the seed network with parallel acceleration. Through continuous simulations, GAG develops diverse graphs by folding the affiliation network based on node types and edge/relation types. Our main contributions are: **(1) Graphs of Real-World Network Structures:** The generated graphs exhibit seven essential structural characteristics observed in real-world networks, including *power-law degree distribution*, *small-world*, *shrinking diameter*, etc. Specifically, GAG surpasses the best-performing baseline by 11% on specific evaluation metrics for graph expansion tasks. **(2) Text-attributed Graph Generation:** In the node classification benchmarking task, GAG demonstrates an average improvement of 1.45 in accuracy retention compared to baseline methods. By effectively capturing the intricate relationship between textual features and graph structures, GAG generates highly realistic text-attributed graphs. **(3) Graph Generation via Large-Scale Agent Simulation:** The framework supports the generation of graphs across ten distinct types, accommodating up to 10 million edges or nearly 100,000 nodes through simulations with up to nearly 100,000 LLM-based agents. Additionally, the parallel acceleration accelerates the simulation with a minimum speed-up of 90.4%.

2 Related Work

Graph Generation As an extensively explored foundational task, existing graph generation methods mainly include two categories: (1) Rule-based methods, which gradually add nodes based on random (Erdos et al., 1960) or preferential attachment (Barabási and Albert, 1999) rules; Though (Barabási and Albert, 1999) successfully models power-law degree distribution, they struggle to capture the community structures prevalent in real-world networks (You et al., 2018). (2) Deep Learning based methods, which aim to capture the complex and diverse real-world structures through training on a network dataset, mainly fall into two categories: Autoregressive methods (You et al., 2018; Dai et al., 2020; Bergmeister et al., 2024) predict edges incrementally for each new node, while one-shot methods (Simonovsky and Komodakis, 2018; De Cao and Kipf, 2018; Liu et al., 2019; Vignac et al., 2023) generate entire graphs in a single step. However, these methods require large-scale training data and struggle to generate graphs outside the training distribution, and fail to generate text-rich attributes. Although some progress has been made with extrapolating to out-of-distribution graphs (Bergmeister et al., 2024), the maximum size of the expanded graph is limited to 144 nodes (Chang et al., 2024). Moreover, they cannot generalize to new graph domains.

LLM-based Human Behavior Simulation With LLMs demonstrating advanced capabilities in human-like responses and autonomous planning (Gao et al., 2023), they are increasingly recognized as a new paradigm for simulations across fields such as education (Chen et al., 2024), social dynamics (Park et al., 2023), and economics (Li et al., 2024b). In graph generation, De Marzo et al. (2023) first explores the scale-free property of power-law distributions in LLM-based agent interactions. Subsequently, Papachristou and Yuan (2024); Chang et al. (2024) examines additional social graph properties. However, these simulations often lack realism due to simplified modeling of human behavior, such as name selection, and are constrained to fewer than 100 agents. Recently, Pan et al. (2024) introduced AgentScope, a framework enabling large-scale multi-agent simulations for simplified human behavior, demonstrated through number-guessing games. Building on this, we have enhanced AgentScope to simulate more complex human behaviors at a large scale.

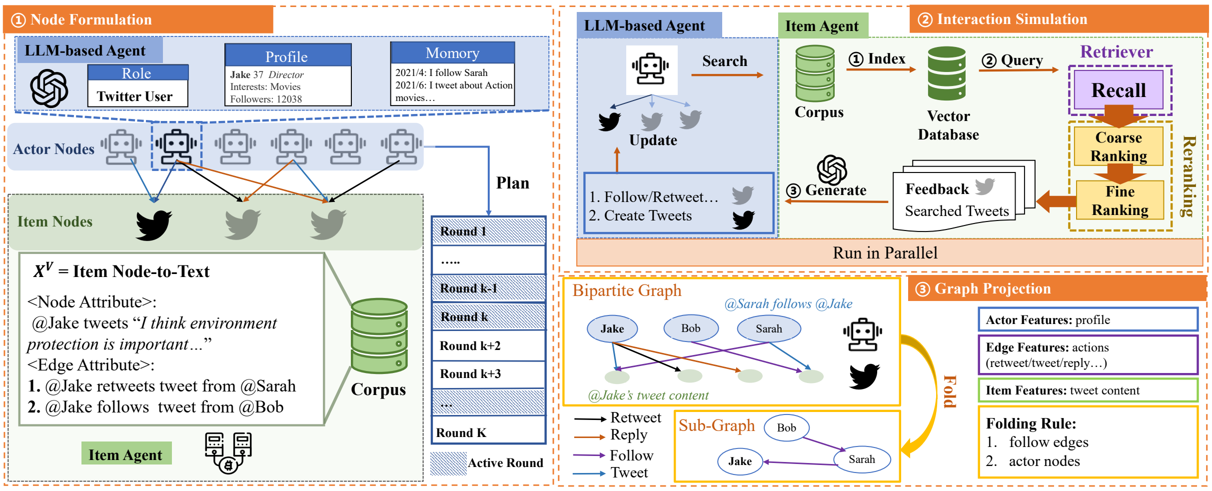


Figure 1: An illustration of the GAG Framework for generating social graphs: (1) **Node Formulation**, where actor and item node sets are initialized; actor nodes are initialized with LLM-based agents, while an item agent manages item nodes; (2) **Interaction Simulation**, where agents engage in pair-wise interactions within a virtual environment; (3) **Graph Projection**, where the actor-item bipartite graph is folded along specified node or edge types.

3 The GAG Framework

In this section, we present GAG, a simulation-based framework designed for large-scale dynamic text-attributed graph generation. GAG aims to eliminate preset rules and training processes in graph generation through simulation-based methods.

3.1 Problem Setup

Graphs serve as mathematical structures that represent interactions among entities (Bergmeister et al., 2024-05). In graph generation research, although numerous methods have been proposed for molecular graph generation, non-molecular graph has received comparatively less attention (Faez et al., 2021). To address this gap, we focus on modeling the dynamic evolution of text-attributed social graphs, which consist of two types of entities: actors and items. The entity-wise interaction can be naturally represented as a bipartite graph $B(\mathcal{A}, \mathcal{V}, \mathcal{E})$, where $\mathcal{A}$ denotes the set of actors, $\mathcal{V}$ denotes the set of items, and $\mathcal{E}$ denotes the edges connecting them (Bergmeister et al., 2024-05). Each vertex in $\mathcal{A}$ and $\mathcal{V}$ is associated with textual features, represented as $\mathcal{X}^A$ for actors and $\mathcal{X}^V$ for items. The goal is to simulate the real-world evolution of B into a larger graph B' over time, where $|\mathcal{A}'| \gg |\mathcal{A}|$, $|\mathcal{V}'| \gg |\mathcal{V}|$, or $|\mathcal{E}'| \gg |\mathcal{E}|$, while preserving graph macroscopic and microscopic properties.

To achieve this, we propose the GAG framework, which simulates actor-item interactions over K simulation rounds. Leveraging the role-playing capabilities of LLMs (Li et al., 2023; Park et al., 2023), we construct n LLM-based agents to form the actor set $\mathcal{A} = \{a_1, a_2, \ldots, a_n\}$, simulating authors, movie watchers, or social media users. Simultaneously, the item set $\mathcal{V} = \{v_1, v_2, \ldots, v_m\}$ is initialized with m item entities, including papers, tweets, or movies, which are managed by an item agent. The simulation begins with a seed graph B_0, B_0 evolves into B_K after K simulation rounds. For the k-th simulation round, $k \in [K]$, the GAG framework employs a three-step simulation workflow, as illustrated in Figure 1:

(1) Node Formulation: Given B_{k-1} as input, the item set V_{k-1} and agent set $\mathcal{A}_{k-1}$ is initialized from B_{k-1}; while an actor sub-set $(\widetilde{\mathcal{A}_k}, \widetilde{X_k^A})$ of pre-set size is initialized for each simulation round:

$$\begin{aligned} \mathcal{A}_k &= \mathcal{A}_{k-1} \cup \widetilde{\mathcal{A}_k}, \\ X_k^A &= X_{k-1}^A \cup \widetilde{X_k^A}. \end{aligned} \tag{1}$$

(2) Interaction Simulation: Agents engage in pairwise interactions within a virtual environment. Each interaction process generate an edge sub-set $\widetilde{E_k}$, and an item sub-set $(\widetilde{V_k}, \widetilde{X_k^V})$:

$$\begin{aligned} \mathcal{V}_k &= \mathcal{V}_{k-1} \cup \widetilde{\mathcal{V}_k}, \\ X_k^V &= X_{k-1}^V \cup \widetilde{X_k^V}, \\ \mathcal{E}_k &= \mathcal{E}_{k-1} \cup \widetilde{\mathcal{E}_k}. \end{aligned} \tag{2}$$

(3) Graph Projection: The resulting graph B_k is folded along specific node or edge types, deriving unipartite or multipartite graphs for analysis.

3.2 Node Formulation

The underlying idea behind GAG is that in social graphs, there are two types of entities — actors and items — that are related by pairwise interactions. Given B_{k-1} as input, to facilitate execution of the k-th simulation round, we first initialize these two node types respectively.

Item Node We first collect the text feature representation of item nodes in B_{k-1}. Specifically, we denote the j-th item node as $v_{j,k-1}$, the textual feature associated as $x^V_{j,k-1}$. To construct $x^V_{j,k-1}$, we leverage an item template that maps each item node to its corresponding real-world textual representation (Feng et al., 2024). This representation includes the node's textual features, with optional incorporation of 1-hop actor-item edge attributes. For example, a paper entity is described using node textual features such as its title, topic, abstract, and neighboring edges, such as the authors writing the paper. Details of the item prompt template are provided in Appendix A.1.

Actor Node The textual features associated with actor nodes in B_{k-1} are denoted as X^A_{k-1}. For the k-th simulation round, LLMs are prompted to generate a pre-set number of role-playing synthetic profiles, denoted as $\widetilde{X^A_k}$. The number is fixed or proportional to the actor set size. These profiles capture various aspects of human personal information, such as research interests, institutional affiliations, and social graph connections. The profile prompt template is detailed in Appendix A.1. Consequently, the textual features of actor nodes in B_k are obtained as: $X^A_k = \{X^A_{k-1} \cup \widetilde{X^A_k}\}$, where $X^A_k = \{x^A_{i,k}, i \in [n]\}$. Given X^A_k, GAG leverages LLM-based agents to instantiate actor nodes. The textual feature $x^A_{i,k}$ forms the characterized profile for $a_{i,k}$. Collectively, these LLM-based agents form the actor node set A_k of the bipartite graph. To enable adaptive learning from past interactions, each LLM-based agent is equipped with a memory component that records its activity history. We organize the memory using reflection (Shinn et al., 2023) and summarization techniques. Moreover, we adopt the action state (active or idle) to control each agent's interaction frequency. In real-world scenarios, human activity often follows a Pareto distribution (Guo et al., 2009), where approximately 20% of users account for 80% of the total activity. We employ two approaches to determine the action state of each agent: 1. A fixed number of A_k is randomly sampled as active agents. 2. We first label the top 20% agents as *core*, while the remaining agents as *regular* based on action history. The agent role labels, combined with action history, are input to the LLM to determine each actor's action state each round.

3.3 Interaction Simulation

Our motivating example is the social networks that emerge from search engine queries (Lattanzi and Sivakumar, 2009). These queries enable actors to filter partial observations from the entire item set. We first simulate this interaction process in a virtual environment, forming the edge set $\widetilde{E_k}$ and a new item set $\widetilde{V_k}$ for each simulation round. To enhance efficiency, we further optimize the simulation process with parallel acceleration.

Interaction Process In real-world scenarios, humans rely on search engines to obtain efficient and targeted environment observations (Yau et al., 2020). Inspired by this, we propose the Simulation-Oriented Retrieval Augmented Generation (S-RAG) framework. For the k-th simulation round, active actor $a_{i,k}$ is provided with an item subset as environment observation, denoted as $O_{i,k}, O_{i,k} \subseteq V_{k-1}$. Following traditional RAG (Cuconasu et al., 2024), S-RAG is divided into three processes as shown in Algorithm 1:

Algorithm 1 S-RAG for the k-th simulation round.

Require: Item Set V_{k-1}, Large Language ModelLLM.
- $\widetilde{V_k} = \emptyset, \widetilde{\mathcal{E}_k} = \emptyset, \widetilde{X^V_k} = \emptyset,$
- **for** $i \in [n]$ **do**
  - **if** $a_{i,k}$ is active **then** $Q_{i,k} = \text{LLM}(a_{i,k} \mid memory),$
  - **else** continue
  - **end if**
  - $O_{i,k} = \emptyset,$
  - **for** $q \in Q_{i,k}$ **do**
    - $O_{i,k,q} = \text{RECALL}(q, V_{k-1}),$
    - $O_{i,k,q} = \text{RERANKING}(O_{i,k,q}, a_{i,k}),$
    - $O_{i,k} = O_{i,k} \cup O_{i,k,q},$
  - **end for**
  - $\widetilde{V_{i,k}}, \widetilde{\mathcal{E}_{i,k}}, \widetilde{X^V_{i,k}}, = \text{LLM}(a_{i,k} \mid O_{i,k}),$
  - $\widetilde{V_k} = \widetilde{V_k} \cup \widetilde{V_{i,k}},$
  - $\widetilde{X^V_k} = \widetilde{X^V_k} \cup \widetilde{X^V_{i,k}},$
  - $\widetilde{\mathcal{E}_k} = \widetilde{\mathcal{E}_k} \cup \mathcal{E}_{i,k},$
- **end for**
- **return** $\widetilde{V_k}, \widetilde{X^V_k}, \widetilde{\mathcal{E}_k}.$

(1) Index Process: Given the item textual features $X^V_{k-1} = \{x^V_{j,k-1}\}, j \in [m]$, where each $x^V_{j,k-1}$ is stored as a text document. We first convert these textual features into a set of embedding vectors E^V_{k-1} using an embedding model $\mathrm{encoder}(\cdot)$. The process involves transforming each $x^V_{j,k-1}$ into a d-dimensional embedding: $e^V_{j,k-1} = \mathrm{encoder}(x^V_{j,k-1}) \in \mathbb{R}^d, j \in [m]$. We store these vectors in a vector database (Douze et al., 2024), managed by the item agent. This process constructs an environment for actors, thereby providing them with item observations.

(2) Query Process: For the actor node $a_{i,k}$ in an active state, it can freely access environmental information. To obtain the most relevant items, the actor first reflects on its memory, which serves as input to the LLM to create a query set $Q_{i,k}$, which contains descriptive keywords. For each query $q \in Q_{i,k}$, we first convert q into an embedding vector: $e_q = \mathrm{encoder}\,(q)$. Next, we specify the desired number of retrieved feedback as N_r and retrieve the top N_r items, denoted as $O_{i,k,q}$. The retrieved feedback for all queries collectively forms the observation: $O_{i,k} = \bigcup_{q \in Q_{i,k}} O_{i,k,q}$. Specifically, the retrieving process of $O_{i,k,q}$ has two stages: 1. In the Recall stage, we filter out the top N_r items by measuring the embedding similarity between X^V_{k-1} and q:

$$O_{i,k,q} = \mathrm{topN_r(q)}_{v_{j,k-1} \in V_{k-1}} \mathrm{Sim}\left(q, x^V_{j,k-1}\right),$$
$$\mathrm{Sim}\left(q, x^V_{j,k-1}\right) = \frac{e_q \cdot e^V_{j,k-1}}{\|e_q\| \cdot \left\|e^V_{j,k-1}\right\|}, j \in [m].$$

2. In the ReRanking stage, we refine and organize $O_{i,k,q}$ according to the attribute of the active actor, which is divided into two phases: (1) Coarse Ranking: Items are reordered to prioritize those created by *core* actors. (2) Fine Ranking: Items are further reordered based on the actor's personal preferences. For example, for an author-actor with expertise in AI, items focused on AI are prioritized in $O_{i,k,q}$. The ReRanking hyperparameters are detailed in Appendix A.2.

(3) Generation Process: In real-world scenarios, based on feedback from search engine queries, the actor acts according to the feedback. For example, in the context of author-paper interaction, the authors may generate a new paper and reference searched papers based on their perception of the environment. To mimic this process, we instruct $a_{i,k}$ using an action template to perform various actions. Each action forms an item-actor edge, where the action type determines the edge label. The action types for different simulation scenarios are listed in Appendix A.2. For creation action, the actor $a_{i,k}$ additionally creates a new item node during the interaction, denoted as $\widetilde{x^V_{i,k}}$ (textual feature) and $\widetilde{v_{i,k}}$ (item node). Moreover, the memory of active actor nodes is updated simultaneously with the action history, further refining their perception of the environment. Across all active agents, the interaction edges collectively form $\widetilde{\mathcal{E}_{i,k}}$, while the created item nodes form $\widetilde{\mathcal{V}_{i,k}}$ and $\widetilde{X^V_{i,k}}$. As a result, the bipartite graph is updated accordingly.

Parallel Acceleration The S-RAG enables the modeling of actor-item interaction processes in real-world scenarios. However, there remains technical barriers in supporting interaction simulation at the scale of $n = 1e^5$. Additionally, we note that the inference time of LLMs is substantial, leading to prolonged IO wait times for idle LLM-based actor agents. Various solutions have been proposed to address this issue, such as async (Kansal, 2024) and actor architecture (Gao et al., 2024). We adopt the parallel processing technique (Gao et al., 2024). As highlighted by (Clauset et al., 2004), network structures often display tightly connected communities with loosely connected inter-community links. In GAG, we categorize agents into distinct groups based on strong internal interactions and weaker inter-group interactions. Specifically, each active actor agent forms a group with the item agent. These groups run in parallel on CPU cores with P ports. The implementation details are in Appendix A.2.

3.4 Graph Projection

B_0 progressively evolves into B_K after K rounds of interaction simulations. For bipartite graphs, different projected subgraphs are folded based on node and edge types. The sub-graph, denoted as $G(\mathcal{V}^s, \mathcal{E}^s)$, evolves via the evolution of B. Textual features associated with $\mathcal{V}^s$ is represented as $\mathcal{X}^{V^s}$. Following established folding rules in network science research, the subgraphs embody different semantic interpretations. For instance, in an author-paper bipartite graph, selecting paper nodes and deriving paper-paper citation edges—i.e., hop-2 edges (paper-author-paper) in the bipartite graph—yields a paper citation network. Folding rules for different domains are provided in Appendix A.3.

4 Experiment

In network science, there has long been an interest in graph structures that emerge within scientific, technological, and sociological contexts (Leskovec et al., 2007). To evaluate our framework, we simulate graph generation across three representative domains: (1) **Scientific Context** (SC): This domain models the dissemination of ideas, theories, and results in science. The simulation involves an author actor set interacting with a paper set, producing citation, bibliographic coupling, co-citation, author citation, and co-authorship networks (Garfield, 2000). Simulation terminates at the citation network reaching 1e4 nodes. (2) **Technological Context** (TC): This domain models customer-product interactions in digital commerce. The simulation involves a user actor set interacting with a movie set, producing movie rating and user projection networks (Zhou et al., 2007). Simulation terminates at the movie rating network reaching 1e5 edges. (3) **Socialogical Context** (SoC): This domain models interpersonal communications in an online social media platform. The simulation involves a user actor set interacting with a tweet item set on platforms like Twitter, producing follow, friend, and action networks (De Domenico et al., 2013). In addition to graph expansion, GAG can also generate graphs from scratch using LLM-generated graph textual features, eliminating the need for external data collection. We employ this method in SoC. Simulation terminates after five rounds.

Evaluation Protocol To evaluate the effectiveness of GAG, we assess three key aspects: First, we compare the generated graph structures with real-world networks at both macro and micro scales. Next, we evaluate the effectiveness of graph textual features with the GNN benchmarking task. Finally, we assess the scalability of GAG in terms of generation scale and efficiency. Details of evaluation setups and metrics are provided in Appendix B.1.

4.1 Graph Structure Alignment

In this paper, we investigate the generated graph structures from both macro and micro perspectives: At the macro level, we examine the graph structure dynamics in the graph evolution and align our observations with established network science theories. At the micro level, we compare GAG to existing graph generation models in capturing the micro-level graph structural characteristics.

Macro-Level Evaluation For macro-level structural characteristic alignment, we examine three structural characteristics observed in real-world networks (Albert and Barabási, 2002): *power-law distribution, small-world phenomenon*, and *shrinking diameter*. Four additional structural characteristics are detailed in Appendix B.2.

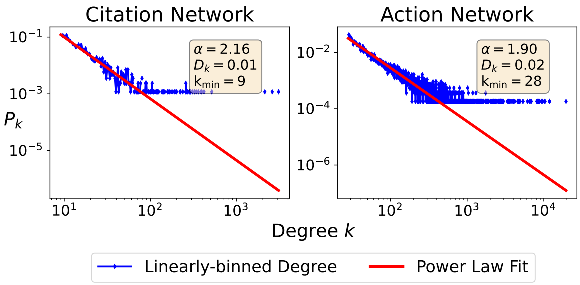


Figure 2: The *power-law* Distribution of degrees in generated graphs. The degree k is plotted against the probability density function P_k on a log-log scale, where α denotes the exponent parameter, k_{min} represents the cut-off k (Alstott et al., 2014).

(1) Power-law Distribution: The degree distribution of scale-free networks often follows a *power-law distribution* (Barabási and Albert, 1999), which is commonly observed in citation networks and social networks. In simulations with GAG, the generated citation, author-citation, and action networks also exhibit this characteristic. We adhere to the established criteria for evaluating whether the network degree distribution follows a power law: $D_k < 0.1$ (Alstott et al., 2014). As shown in Figure 2, the degree distribution of these networks follow a *power-law* distribution with exponent parameter (Clauset et al., 2009) $\alpha \in [1.90, 2.16]$.

(2) Small World Phenomenon: Real-world networks exhibit a *small world* phenomenon (Mislove et al., 2007; Watts and Strogatz, 1998), characterized by a small diameter and a high clustering coefficient. Table 1 compares the $\bar{cc}$ of the generated graphs with that of the random graphs with consistent average degree: Erdős-Rényi (Erdos et al., 1960) and Barabási-Albert graphs (Barabási and Albert, 1999). The results indicate that generated social graphs (i.e., follow, friend, and action networks), exhibit a significantly higher $\bar{cc}$ than those of the random graphs, confirming these networks exhibit small-world characteristics. Additionally, the generated networks exhibit a small diameter, $D_e \in [1.17, 11.66]$, consistent with the *six degrees of separation phenomenon* observed in real-world networks (Leskovec et al., 2007; Broder et al., 2000). The details of the structural metrics for generated graphs are listed in Appendix B.1. The high $\bar{cc}$, combined with small D_e, confirms these networks exhibit small-world characteristics.

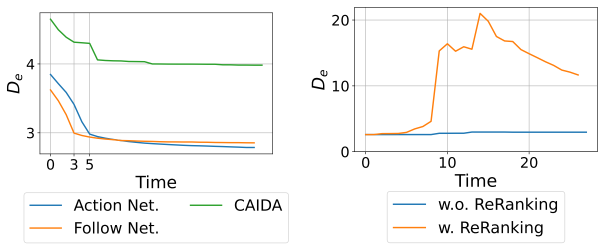
Table 1: $\bar{cc}$ of the generated networks, and the ratio to Erdös-Rényi and Barabási-Albert graph models. A dash (—) signifies that $\bar{cc} = 0$ for the graph model. The results indicate that generated social graphs exhibit the small-world characteristic.

	Graph Scale		**Ratio to Random Graphs**	
	$\lvert\mathcal{V}^s\rvert$	$\lvert\mathcal{E}^s\rvert$	ER	BA
Paper Citation	1.14e+04	3.63e+04	301.08	—
Bib-Coupling	1.09e+04	1.22e+07	7.46	4.40
Co-Citation	3.93e+03	3.27e+04	275.97	44.87
Author Citation	5.01e+03	2.41e+05	39.82	11.19
Co-Authorship	5.01e+03	2.08e+04	234.81	17.59
Action	9.97e+04	9.07e+05	784.93	73.97
Follow	9.96e+04	1.53e+06	3961.83	443.80
Friend	9.96e+04	5.01e+05	19768.58	1391.47
Movie Rating	4.17e+03	3.25e+04	0.00	0.00
User Projection	3.91e+03	9.04e+05	5.78	2.82

(3) Shrinking Diameter: The *shrinking diameter* is a notable phenomenon in social graphs (Leskovec et al., 2007), with D_e decreasing as the network evolves over time. We construct SoC with $N = 7000$, and investigate the graph evolution processes of follow, friend, and action networks for 30 simulation rounds. We calculate the effective diameter D_e for both the generated graphs and the real-world network: CAIDA[1]. As shown in Figure 3a, D_e decreases at a slow pace, identical to the trend observed in (Leskovec et al., 2007) and CAIDA. To explain *shrinking diameter*, the Forest Fire model (Leskovec et al., 2007) employs a modified preferential attachment mechanism, referred to as community-guided attachment. In GAG, the ReRanking process enhances personalized recommendations. We conduct an ablation experiment to assess the effect of the ReRanking. As shown in Figure 3b, we observe an initial increase in D_e followed by a rapid decline. Notably, upon removing the ReRanking, the D_e trends upwards from 2.6 to 2.96, indicating that ReRanking fosters community-guided attachment.

Micro-Level Evaluation We observe that real-world social networks tend to expand over time and gradually become denser, often exhibiting a power-law degree distribution (Clauset et al., 2009). For micro-level structural alignment, to assess whether GAG can simulate the temporal evolution of a network, from a small seed graph to a larger structure, while preserving key structural properties, we evaluate its performance on the expansion process of a citation network. For dataset, specifically, we partition the CiteSeer network (Sen et al., 2008) into $G_{<t}$ and $G_{>t}$ based on a temporal threshold t, where small train subgraphs are sampled from $G_{<t}$ and large test subgraphs from $G_{>t}$. We compare GAG against existing graph generation models. Since baseline deep learning-based graph generators are trained on fixed-size graphs, we adapt their sampling procedures to support extrapolation to unseen graph sizes (Bergmeister et al., 2024).

In our setup, we select the first graph from the training set as the seed graph input to GAG. For deep learning-based baselines, models are trained on the whole training dataset and generate graphs matching the size of the test graphs. For random graph baselines, we precompute the required hyper-parameters based on the degree distribution of the complete CiteSeer network. We set up the Valid metric to test the proportion of generated graphs that follow a power-law distribution and report the Kolmogorov–Smirnov statistic D_k to quantify power-law fitness. Details of datasets and metrics are provided in Appendix B.3. As shown in Table 2, apart from the Barabási-Albert and GN (Krapivsky and Redner, 2001) model, the graphs generated by GAG also adhere to a *power-law* distribution with $\alpha = 2.39$. In contrast, deep-learning-based models tend to overfit the seed graph and fail to generate graphs that accurately follow the *power-law* distribution. Surprisingly, GAG outperforms most baseline models on MMD metrics, which is notable given that no explicit graph structure constraints are imposed on the LLM-based agents during edge creation. Additionally, we conducted comparative experiments on the seed graphs across different



(a) w.ReRanking (b) w.o. ReRanking

Figure 3: The *Shrinking Diameter* phenomenon simulated by GAG; The left figure demonstrates that as the graph evolves, D_e gradually decreases in action and follows networks; The right figure presents an ablation experiment of the ReRanking, demonstrating its effect on D_e in friend network.

[1] https://sparse.tamu.edu/SNAP/as-caida

Table 2: Comparison with existing graph generation models for the graph expansion task. For GRAN and GraphMaker, the generated graphs fail to converge to a *power-law* distribution.

	MMD.D↓	MMD.C↓	MMD.S↓	MMD.O↓	D_k	α	Valid↑	GEM
CiteSeer	-	-	-	-	$0.06_{\pm 0.0}$	$2.38_{\pm 0.0}$	1.0	-
Erdös-Rényi	0.26	1.41	0.57	1.41	$0.1_{\pm 0.01}$	$3.72_{\pm 0.13}$	0.0	0.29
Barabási-Albert	**0.19**	1.41	**0.27**	1.08	$0.04_{\pm 0.01}$	$2.38_{\pm 0.04}$	**1.0**	0.47
Small-World	0.72	1.38	0.60	1.41	$0.42_{\pm 0.01}$	$2.03_{\pm 0.0}$	0.0	0.22
SBM	0.88	1.19	0.85	1.02	$0.60_{\pm 0.00}$	$1.24_{\pm 0.00}$	0.0	0.22
GN	<u>0.20</u>	1.41	0.26	1.10	$0.04_{\pm 0.01}$	$2.38_{\pm 0.02}$	**1.0**	0.47
BiGG	0.63	1.13	0.65	1.23	$0.27_{\pm 0.01}$	$1.69_{\pm 0.01}$	0.0	0.23
GRAN	0.36	<u>0.55</u>	0.72	1.41	-	$4.16_{\pm 0.39}$	0.0	0.26
BwR	0.49	1.41	0.66	1.41	$0.07_{\pm 0.09}$	$4.46_{\pm 0.02}$	0.0	0.22
GraphMaker	0.47	1.41	0.83	1.41	-	-	0.0	0.22
L-PPGN	0.76	1.19	0.78	<u>1.05</u>	$0.39_{\pm 0.03}$	$1.36_{\pm 0.02}$	0.0	0.23
GAG	**0.16**	**0.19**	<u>0.32</u>	**1.02**	$0.08_{\pm 0.01}$	$2.37_{\pm 0.03}$	**1.0**	0.52

datasets. Regarding GEM, GAG surpasses the best-performing baseline by 11% (0.52 vs. 0.47). This suggests that GAG effectively simulates human behavior patterns, generating graphs that closely resemble the structural characteristics of real-world networks in graph expansion. Moreover, we found that deep learning models benefit from denser seed graphs, which provide strong structural priors for generating out-of-distribution graphs. Additional experimental details are provided in Appendix B.3.

4.2 Textual Feature Alignment

GAG generates text-rich dynamic graphs by collecting actor-item interaction data, simulating the process of distilling textual information into structured representations that resemble real-world networks. To evaluate whether the generated graphs preserve the text-structure correlations of the seed graph, we adopt Graph Neural Network (GNNs) benchmarking task of node classification (Yoon et al., 2023). Specifically, we train different GNN models on both the generated graph and the seed graph, measuring the accuracy gap between the two, denoted as ΔAcc. We apply Bert embedding (Reimers and Gurevych, 2019) to embed textual features into vectors for GNN training. A smaller ΔAcc indicates better preservation of the text-structure correlations. For benchmarking, we select four representative GNN models. For the baseline graph generation models, we randomly assign text features generated by LLMs to the nodes. Details of the experimental setup are provided in Appendix C. As shown in Table 3, the random baselines for both structure and node feature perform the worst. Existing models such as GRAN and GraphMaker demonstrate strong performance in GNN benchmarking tasks (Li et al., 2024a). However, GAG consistently outperforms these baselines, achieving an average improvement of 1.45 in ΔAcc across GNNs, with ΔAcc values ranging from 0.09 to 3.61. These results demonstrate GAG's effectiveness in capturing intricate text-structure correlations.

Table 3: Performance gap between seed graph and generated graph for different graph generation models on the node classification task, lower gap means better generation quality.

	$\Delta ACC\downarrow$			
	GAT	GCN	GCNII	GraphSage
SF.random	$13.4_{\pm 3.5}$	$15.0_{\pm 2.0}$	$9.6_{\pm 0.9}$	$7.0_{\pm 0.9}$
F.random	$24.1_{\pm 2.8}$	$23.2_{\pm 2.3}$	$9.3_{\pm 1.4}$	$7.1_{\pm 1.7}$
S.random	$18.9_{\pm 2.5}$	$21.8_{\pm 2.0}$	$2.2_{\pm 1.6}$	$3.2_{\pm 1.8}$
BiGG.L	$39.4_{\pm 4.5}$	$34.1_{\pm 6.6}$	$4.7_{\pm 4.4}$	$3.4_{\pm 3.5}$
GRAN.L	$5.3_{\pm 3.6}$	$6.1_{\pm 5.1}$	$3.5_{\pm 3.4}$	$4.4_{\pm 2.5}$
BwR.L	$35.3_{\pm 2.9}$	$38.5_{\pm 4.7}$	$4.8_{\pm 4.5}$	$7.6_{\pm 4.3}$
GraphMaker.L	$4.0_{\pm 3.5}$	$\mathbf{2.8}_{\pm 4.0}$	$3.6_{\pm 4.2}$	$4.17_{\pm 4.03}$
L-PPGN.L	$38.4_{\pm 6.4}$	$32.6_{\pm 3.5}$	$4_{\pm 3.6}$	$3.9_{\pm 3.8}$
GAG	$\mathbf{2.3}_{\pm 1.2}$	$3.6_{\pm 1.3}$	$\mathbf{0.5}_{\pm 1.5}$	$\mathbf{0.1}_{\pm 1.7}$

4.3 Scalability of GAG

We assess GAG scalability in terms of both graph generation scale and efficiency. Regarding generation scale, as shown in Table 1, GAG supports large-scale graph generation of up to nearly 100,000 nodes in the action network, or 12.2 million edges in the bib-coupling network. In contrast, existing graph generation models are limited to 5,000 nodes (Bergmeister et al., 2024; Liao et al., 2019) or sparse grid graphs (Dai et al., 2020); the detailed comparison with existing graph generation models is presented in Appendix D. Regarding generation efficiency, we evaluate the effectiveness of parallel acceleration by examining the impact of P on the runtime performance of the GAG framework. Experiments are conducted on a machine equipped with 96 CPU cores and 376GB of memory. As shown in Table 4, when the number of actor agents (n) is held constant, the time required to generate one item-actor interaction data is reduced by at least 97.5% when $P > 1$ compared to $P = 1$. This highlights the effectiveness of parallel acceleration and the capability of GAG in supporting the simulation of large-scale graph evolution.

Table 4: The time cost (*min*) of 40 actor agents to interact once with the item agent.

P	SC	TC	SoC
1	3.6250	0.0683	0.0623
4	0.1470	0.0068	0.0112
16	0.1160	0.0053	0.0109
24	0.0910	0.0054	0.0060
1→24	**↓97.5%**	**↓92.1%**	**↓90.4%**

5 Conclusion

In this study, we present GAG, a novel and general framework designed for generating dynamic, large-scale, text-rich graphs with human interaction simulation. The generated graphs exhibit seven macro-level characteristics of real-world networks, including power law, small world, and shrinking diameter. In the graph expansion task, GAG surpasses existing baselines in graph expansion tasks by 11% on specific evaluation metrics. Furthermore, we present the S-RAG algorithm for simulating diverse human interaction processes at scale, complemented by parallel acceleration for simulation speed-up, achieving a speed-up of at least 90.4%. Our framework successfully produces high-quality graphs with up to nearly 100,000 nodes or 10 million edges. Overall, GAG represents a promising initial step toward the efficient generation of dynamic, large-scale, text-rich graphs.

Limitations

This paper acknowledges several limitations that future research could address:

Behavior Interpretability Though previous work has demonstrated that persona-enhanced prompting effectively guides LLMs in generating distinctive role-play synthetic data (Chan et al., 2024), the mechanism of in-context learning remains a black box. Specifically, it remains unclear which prompts instruct agents to exhibit heterogeneous behavior, a challenge in LLM-based simulation works. Existing approaches explore the prompt-behavior correlation in LLMs through Knowledge Circuit (Yao et al., 2024b) or SAE-based representation engineering (Zhao et al., 2024). In the future, we aim to integrate such methods to explain the diverse behaviors of LLM-based agents.

Simulation Scenario We acknowledge that simulation-based graph generation is currently suitable only for social networks. For domains such as point clouds, traffic networks, and molecular graphs, GAG is not directly applicable. However, as LLMs store a significant amount of factual knowledge in their parameters, we believe that with the continued development of LLM capabilities, the simulation of LLM-based agents can be extended to other domains.

Ethics Statement

This work fully complies with the ACL Ethics Policy. To the best of our knowledge, we declare that there are no ethical issues in this paper.

Acknowledgement

This research was supported in part by National Natural Science Foundation of China (No. 92470128, No. U2241212), by Beijing Outstanding Young Scientist Program No.BJJWZYJH012019100020098, by Alibaba Group through Alibaba Innovative Research Program. We also wish to acknowledge the support provided by the fund for building world-class universities (disciplines) of Renmin University of China, by Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education, by Intelligent Social Governance Interdisciplinary Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Public Policy and Decision-making Research Lab, and Public Computing Cloud, Renmin University of China.

References

Sungsoo Ahn, Binghong Chen, Tianzhe Wang, and Le Song. 2021. Spanning tree-based graph generation for molecules. In *International Conference on Learning Representations*.

AI@Meta. 2024. Llama 3 model card.

Réka Albert and Albert-László Barabási. 2002. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47.

Jeff Alstott, Ed Bullmore, and Dietmar Plenz. 2014. powerlaw: a python package for analysis of heavy-tailed distributions. *PloS one*, 9(1):e85777.

Albert-László Barabási and Réka Albert. 1999. Emergence of scaling in random networks. *science*, 286(5439):509–512.

Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. 2024. Efficient and scalable graph generation through iterative local expansion. In *The Twelfth International Conference on Learning Representations*.

Andreas Bergmeister, Karolis Martinkus, Nathanaël Perraudin, and Roger Wattenhofer. 2024-05. Efficient and scalable graph generation through iterative local expansion. s.l. OpenReview. 12th International Conference on Learning Representations (ICLR 2024); Conference Location: Vienna, Austria; Conference Date: May 7-11, 2024; Poster presentation.

Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. 2000. Graph structure in the web. *Computer Networks*, 33(1):309–320.

Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. 2024. Scaling synthetic data creation with 1,000,000,000 personas. *Preprint*, arXiv:2406.20094.

Serina Chang, Alicja Chaszczewicz, Emma Wang, Maya Josifovska, Emma Pierson, and Jure Leskovec. 2024. Llms generate structurally realistic social networks but overestimate political homophily. *Preprint*, arXiv:2408.16629.

Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. 2024. Agentverse: Facilitating multi-agent collaboration and exploring emergent behaviors. In *The Twelfth International Conference on Learning Representations*.

Aaron Clauset, Mark EJ Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 70(6):066111.

Aaron Clauset, Cosma Rohilla Shalizi, and Mark EJ Newman. 2009. Power-law distributions in empirical data. *SIAM review*, 51(4):661–703.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR 2024. ACM.

Hanjun Dai, Azade Nazi, Yujia Li, Bo Dai, and Dale Schuurmans. 2020. Scalable deep generative modeling for sparse graphs. In *International conference on machine learning*, pages 2302–2312. PMLR.

Nicola De Cao and Thomas Kipf. 2018. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*.

Manlio De Domenico, Antonio Lima, Paul Mougel, and Mirco Musolesi. 2013. The anatomy of a scientific rumor. *Scientific reports*, 3(1):2980.

Giordano De Marzo, Luciano Pietronero, and David Garcia. 2023. Emergence of scale-free networks in social interactions among large language models. *arXiv preprint arXiv:2312.06619*.

Nathaniel Lee Diamant, Alex M Tseng, Kangway V Chuang, Tommaso Biancalani, and Gabriele Scalia. 2023. Improving graph generation by restricting graph bandwidth. In *International Conference on Machine Learning*, pages 7939–7959. PMLR.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. 2024. The faiss library. *Preprint*, arXiv:2401.08281.

Yuanqi Du, Xiaojie Guo, Hengning Cao, Yanfang Ye, and Liang Zhao. 2022a. Disentangled spatiotemporal graph generative models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6541–6549.

Yuanqi Du, Xiaojie Guo, Amarda Shehu, and Liang Zhao. 2022b. Interpretable molecular graph generation via monotonic constraints. In *Proceedings of the 2022 SIAM International Conference on Data Mining (SDM)*, pages 73–81. SIAM.

Yuanqi Du, Xiaojie Guo, Yinkai Wang, Amarda Shehu, and Liang Zhao. 2022c. Small molecule generation via disentangled representation learning. *Bioinformatics*, 38(12):3200–3208.

Paul Erdos, Alfréd Rényi, et al. 1960. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 5(1):17–60.

Faezeh Faez, Yassaman Ommi, Mahdieh Soleymani Baghshah, and Hamid R. Rabiee. 2021. Deep graph generators: A survey. *IEEE Access*, 9:106675–106702.

Jiarui Feng, Hao Liu, Lecheng Kong, Yixin Chen, and Muhan Zhang. 2024. Taglas: An atlas of text-attributed graph datasets in the era of large graph and language models. *Preprint*, arXiv:2406.14683.

James H. Fowler and Nicholas A. Christakis. 2010. Cooperative behavior cascades in human social networks. *Proceedings of the National Academy of Sciences*, 107(12):5334–5338.

Chen Gao, Xiaochong Lan, Nian Li, Yuan Yuan, Jingtao Ding, Zhilun Zhou, Fengli Xu, and Yong Li. 2023. Large language models empowered agent-based modeling and simulation: A survey and perspectives. *arXiv preprint arXiv:2312.11970*.

Dawei Gao, Zitao Li, Weirui Kuang, Xuchen Pan, Daoyuan Chen, Zhijian Ma, Bingchen Qian, Liuyi Yao, Lin Zhu, Chen Cheng, et al. 2024. Agentscope: A flexible yet robust multi-agent platform. *arXiv preprint arXiv:2402.14034*.

Eugene Garfield. 2000. *The web of knowledge: a festschrift in honor of Eugene Garfield*. Information Today, Inc.

Lei Guo, Enhua Tan, Songqing Chen, Xiaodong Zhang, and Yihong Zhao. 2009. Analyzing patterns of user content generation in online social networks. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 369–378.

Shubham Gupta, Sahil Manchanda, Srikanta Bedathur, and Sayan Ranu. 2022. Tigger: Scalable generative modelling for temporal interaction graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 6819–6828.

Jiaqi Han, Jiacheng Cen, Liming Wu, Zongzhao Li, Xiangzhe Kong, Rui Jiao, Ziyang Yu, Tingyang Xu, Fandi Wu, Zihe Wang, et al. 2024. A survey of geometric graph neural networks: Data structures, models and applications. *arXiv preprint arXiv:2403.00485*.

F. Maxwell Harper, Joseph A. Konstan, and Joseph A. 2016. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5:19:1–19:19.

Carl Hewitt, Peter Bishop, and Richard Steiger. 1973. A universal modular actor formalism for artificial intelligence. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, IJCAI'73, page 235–245, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Nathan O. Hodas, Farshad Kooti, and Kristina Lerman. 2013. Friendship paradox redux: Your friends are more interesting than you. *Preprint*, arXiv:1304.3480.

Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. 1983. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137.

Shion Honda, Hirotaka Akita, Katsuhiko Ishiguro, Toshiki Nakanishi, and Kenta Oono. 2019. Graph residual flow for molecular graph generation. *arXiv preprint arXiv:1909.13521*.

Wengong Jin, Regina Barzilay, and Tommi Jaakkola. 2018. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR.

Jaehyeong Jo, Seul Lee, and Sung Ju Hwang. 2022. Score-based generative modeling of graphs via the system of stochastic differential equations. In *International conference on machine learning*, pages 10362–10383. PMLR.

Don H Johnson. 2006. Signal-to-noise ratio. *Scholarpedia*, 1(12):2088.

Aarushi Kansal. 2024. *LangChain: Your Swiss Army Knife*, pages 17–40. Apress, Berkeley, CA.

Wataru Kawai, Yusuke Mukuta, and Tatsuya Harada. 2019. Scalable generative models for graphs with graph attention mechanism. *arXiv preprint arXiv:1906.01861*.

Mahdi Khodayar, Jianhui Wang, and Zhaoyu Wang. 2019. Deep generative graph distribution learning for synthetic power grids. *arXiv preprint arXiv:1901.09674*.

Paul L Krapivsky and Sidney Redner. 2001. Organization of growing random networks. *Physical Review E*, 63(6):066123.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Silvio Lattanzi and D. Sivakumar. 2009. Affiliation networks. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 427–434, New York, NY, USA. Association for Computing Machinery.

Seul Lee, Jaehyeong Jo, and Sung Ju Hwang. 2023. Exploring chemical space with score-based out-of-distribution generation. In *International Conference on Machine Learning*, pages 18872–18892. PMLR.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM Trans. Knowl. Discov. Data*, 1(1):2–es.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. Camel: Communicative agents for "mind" exploration of large language model society. NIPS '23.

Mufei Li, Eleonora Kreačić, Vamsi K. Potluru, and Pan Li. 2024a. Graphmaker: Can diffusion models generate large attributed graphs? *Transactions on Machine Learning Research*.

Nian Li, Chen Gao, Mingyu Li, Yong Li, and Qingmin Liao. 2024b. EconAgent: Large language model-empowered agents for simulating macroeconomic activities. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15523–15536, Bangkok, Thailand. Association for Computational Linguistics.

Renjie Liao, Yujia Li, Yang Song, Shenlong Wang, Will Hamilton, David K Duvenaud, Raquel Urtasun, and Richard Zemel. 2019. Efficient graph generation with graph recurrent attention networks. *Advances in neural information processing systems*, 32.

Jenny Liu, Aviral Kumar, Jimmy Ba, Jamie Kiros, and Kevin Swersky. 2019. Graph normalizing flows. *Advances in Neural Information Processing Systems*, 32.

Meng Liu, Keqiang Yan, Bora Oztekin, and Shuiwang Ji. 2021. Graphebm: Molecular graph generation with energy-based models. *arXiv preprint arXiv:2102.00546*.

Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander Gaunt. 2018. Constrained graph variational autoencoders for molecule design. *Advances in neural information processing systems*, 31.

Youzhi Luo, Keqiang Yan, and Shuiwang Ji. 2021. Graphdf: A discrete flow model for molecular graph generation. In *International conference on machine learning*, pages 7192–7203. PMLR.

Kaushalya Madhawa, Katsuhiko Ishiguro, Kosuke Nakago, and Motoki Abe. 2019. Graphnvp: An invertible flow-based model for generating molecular graphs.

Karolis Martinkus, Andreas Loukas, Nathanaël Perraudin, and Roger Wattenhofer. 2022. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *International Conference on Machine Learning*, pages 15159–15179. PMLR.

Łukasz Maziarka, Agnieszka Pocha, Jan Kaczmarczyk, Krzysztof Rataj, Tomasz Danel, and Michał Warchoł. 2020. Mol-cyclegan: a generative model for molecular optimization. *Journal of Cheminformatics*, 12(1):2.

Alan Mislove, Massimiliano Marcon, Krishna P Gummadi, Peter Druschel, and Bobby Bhattacharjee. 2007. Measurement and analysis of online social networks. In *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pages 29–42.

Xinyi Mou, Zhongyu Wei, and Xuanjing Huang. 2024. Unveiling the truth and facilitating change: Towards agent-based large-scale social movement simulation. *arXiv preprint arXiv:2402.16333*.

M. E. J. Newman. 2002. Assortative mixing in networks. *Phys. Rev. Lett.*, 89:208701.

Chenhao Niu, Yang Song, Jiaming Song, Shengjia Zhao, Aditya Grover, and Stefano Ermon. 2020. Permutation invariant graph generation via score-based generative modeling. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 4474–4484. PMLR.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Xuchen Pan, Dawei Gao, Yuexiang Xie, Zhewei Wei, Yaliang Li, Bolin Ding, Ji-Rong Wen, and Jingren Zhou. 2024. Very large-scale multi-agent simulation in agentscope. *arXiv preprint arXiv:2407.17789*.

Marios Papachristou and Yuan Yuan. 2024. Network formation and dynamics among multi-llms. *arXiv preprint arXiv:2402.10659*.

Joon Sung Park, Joseph O'Brien, Carrie Jun Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*, UIST '23, New York, NY, USA. Association for Computing Machinery.

Marco Podda, Davide Bacciu, and Alessio Micheli. 2020. A deep generative model for fragment-based molecule generation. In *International conference on artificial intelligence and statistics*, pages 2240–2250. PMLR.

Mariya Popova, Mykhailo Shvets, Junier Oliva, and Olexandr Isayev. 2019. Molecularrnn: Generating realistic molecular graphs with optimized properties. *arXiv preprint arXiv:1905.13372*.

Filippo Radicchi, Santo Fortunato, and Alessandro Vespignani. 2011. Citation networks. *Models of science dynamics: Encounters between complexity theory and information sciences*, pages 233–257.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective classification in network data. *AI Mag.*, 29(3):93–106.

Chence Shi, Minkai Xu, Zhaocheng Zhu, Weinan Zhang, Ming Zhang, and Jian Tang. 2020. Graphaf: a flow-based autoregressive model for molecular graph generation. *arXiv preprint arXiv:2001.09382*.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *arXiv preprint arXiv:2303.11366*.

Martin Simonovsky and Nikos Komodakis. 2018. Graphvae: Towards generation of small graphs using variational autoencoders. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27*, pages 412–422. Springer.

Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. 2011. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*.

Clement Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. Digress: Discrete denoising diffusion for graph generation. In *The Eleventh International Conference on Learning Representations*.

Clément Vignac, Igor Krawczuk, Antoine Siraudin, Bohan Wang, Volkan Cevher, and Pascal Frossard. 2023. Digress: Discrete denoising diffusion for graph generation. In *Proceedings of the 11th International Conference on Learning Representations*.

Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of 'small-world' networks. *nature*, 393(6684):440–442.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Yang Yao, Xin Wang, Zeyang Zhang, Yijian Qin, Ziwei Zhang, Xu Chu, Yuekui Yang, Wenwu Zhu, and Hong Mei. 2024a. Exploring the potential of large language models in graph generation. *arXiv preprint arXiv:2403.14358*.

Yunzhi Yao, Ningyu Zhang, Zekun Xi, Mengru Wang, Ziwen Xu, Shumin Deng, and Huajun Chen. 2024b. Knowledge circuits in pretrained transformers. *arXiv e-prints*, pages arXiv–2405.

Y Yau, M Dadar, M Taylor, Y Zeighami, L K Fellows, P Cisek, and A Dagher. 2020. Neural Correlates of Evidence and Urgency During Human Perceptual Decision-Making in Dynamically Changing Conditions. *Cerebral Cortex*, 30(10):5471–5483.

Minji Yoon, Yue Wu, John Palowitch, Bryan Perozzi, and Russ Salakhutdinov. 2023. Graph generative model for benchmarking graph neural networks. In *Proceedings of the 40th International Conference on Machine Learning*, pages 40175–40198.

Jiaxuan You, Rex Ying, Xiang Ren, William Hamilton, and Jure Leskovec. 2018. Graphrnn: Generating realistic graphs with deep auto-regressive models. In *International conference on machine learning*, pages 5708–5717. PMLR.

Zhe Yuan, Zhewei Wei, Fangrui Lv, and Ji-Rong Wen. 2024. Index-free triangle-based graph local clustering. *Frontiers of Computer Science*, 18(3):183404.

Chengxi Zang and Fei Wang. 2020. Moflow: an invertible flow model for generating molecular graphs. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 617–626.

Yu Zhao, Alessio Devoto, Giwon Hong, Xiaotang Du, Aryo Pradipta Gema, Hongru Wang, Kam-Fai Wong, and Pasquale Minervini. 2024. Steering knowledge selection behaviours in llms via sae-based representation engineering. *arXiv preprint arXiv:2410.15999*.

Yanping Zheng, Lu Yi, and Zhewei Wei. 2025. A survey of dynamic graph neural networks. *Frontiers of Computer Science*, 19(6):1–18.

Tao Zhou, Jie Ren, Matúš Medo, and Yi-Cheng Zhang. 2007. Bipartite network projection and personal recommendation. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 76(4):046115.

A Details of GAG

To demonstrate the versatility of the GAG Framework, we build three graph generation tasks for different human activities in our experiments. The concrete experiment settings are as follows: (1) SC: In this scenario, LLM-based agents act as authors interacting with a paper database and generate the following networks: paper citation, bibliographic coupling (bib-coupling), co-citation, author citation, and co-authorship networks (Garfield, 2000). (2) TC: In this scenario, LLM-based agents act as reviewers, interacting with a movie database and generating the following networks: the movie rating and the user projection networks (Zhou et al., 2007). (3) SoC: In this scenario, LLM-based agents act as users interacting with a Twitter-like social media database and generate the following networks: follow, friend, and action networks (De Domenico et al., 2013).

A.1 Node Formulation

We initialize the original item and actor node set from B_0. We use Citeseer as B_0 for SC and Movielens as B_0 for TC. For SoC, we use LLM-generated graph textual features to construct the initial item and actor node set, eliminating the need for external data collection. Further details regarding the configuration of the actors and items are summarized in Table 5.

Item Node Since the seed graph B_0 lacks certain text-rich item nodes and actor nodes (e.g., the Citeseer dataset is missing author information and article content), we crawl the necessary node attributes to enrich the text attributes for Citeseer (Sen et al., 2008). The text-enriched dataset is available in the open-source repository.

For the k-th ($k > 1$) simulation round, we employ an item template that maps each item node $v_{j,k-1}$ to the corresponding text attribute vector $x^V_{j,k-1}$ (Feng et al., 2024). This graph-to-text transformation process leverages the node features and optionally considers edge features to enhance the representation. For instance, in a citation network, graph nodes represent academic papers, whereas in a movie-rating network, they represent movies. The item templates for each simulation scenario are detailed below:

Table 6: The item template of papers in SC.

Node Feature: Academic paper. Title: <Title> Topic: <Topic> Abstract: <Abstract> **Edge Feature**: The citation/writing relationship connecting papers and authors.

Table 7: The item template of movies in TC.

Node Feature (v_i, x_i): Movie. Title: <Title> Genres: <Genres> Content: <Movie Abstract> **Edge Feature**: The movie rating data connecting watchers and movies.

Table 8: The item template of tweets in SoC.

Node Feature (v_i, x_i): Tweets. Tweet ID: <Tweet ID> User: <Tweet User> Tweet: <Tweet Content> **Edge Feature**: The tweet history connecting tweets and tweet users.

Actor Node In the k-th simulation round, we generate a specified number of role-playing synthetic profiles, referred to as $\widetilde{X_k^A}$. For the various simulation scenarios, we develop LLM-based agents to form the actor node set, each assigned distinct roles such as paper authors, movie watchers, or Twitter users. These agents engage with an item set through a predefined set of actions.

Each round includes the addition of profiles, with 30 actor profiles for the SC simulation and 25 for the SoC simulation. For the TC simulation, actor profiles are dynamically added based on node timestamp information from the Movielens. The prompts used for profile generation are outlined in Table 22, Table 23, Table 24.

To determine the action state of each agent, we employ two distinct strategies. For the SC simulation, the number of active agents is proportional to

Table 5: The type of actors and items for different simulation scenarios.

Scenario	Seed Graph	Actor Type	Item Type	Action Type
SC	Citeseer (Sen et al., 2008), Cora (Sen et al., 2008) LLM-Generated	Paper Author	Papers	Creation, Citation
TC	Movielens (Harper et al., 2016), LLM-Generated	Movie Watcher	Movies	Rating
SoC	LLM-Generated	Twitter User	Tweets	Tweet, Reply, Follow

the number of papers created in that round (set at 50). For the TC simulation, all actors remain active. For the SoC simulation, we use a different approach. Since in online social media networks, the influence of content shared can vary significantly between core users (those with a higher level of engagement or influence) and general users. Research indicates that core users typically make up around 20% of the entire user base in a social graph, following the Pareto distribution principle (Mislove et al., 2007; Mou et al., 2024). We categorize the core users as agents labeled as *core*, denoted as HUB. This characterization allows us to analyze the dynamics of influence within simulated environments. We further explore variations by adjusting the ratio of LLM-based agents designated as *core* in Appendix E.2.

A.2 Interaction Simulation

Interaction Process In the k-th simulation round, to identify the most relevant information for $a_{i,k}$, we propose the S-RAG algorithm. As the actor is prompted to give the initial query set $Q_{i,k}$, we collect the feedback $O_{i,k,q}$ for every query $q \in Q_{i,k}$. These feedbacks collectively form the environment feedback set $O_{i,k}$. For a detailed explanation of the query process for one query to obtain $O_{i,k,q}$ in S-RAG:

1. In the recall stage, we initially retrieve $O_{i,k,q}$ as the candidate documents, which serves as the initial environmental feedback. In this step, we filter out N_r candidate documents.

2. In order to align $O_{i,k,q}$ with the agent's personal preference more accurately, we adopt the reranking stage for post-processing of $O_{i,k,q}$, which is divided into two phases: (1) Coarse Ranking: We reorder $O_{i,k,q}$ based on whether the interaction data was generated by agents labeled as *core*. Candidate documents originating from *core* agents are positioned at the forefront of $O_{i,k,q}$, while those from non-core agents are placed towards the end. (2) Fine Ranking: We further reorganize $O_{i,k,q}$ based on the personal preferences of the agents. For SC simulation, the filter items include topics of the academic papers; for TC simulation, the filter items include movie genres; and for SoC simulation, the filter items include attributes of posted tweets, such as friends, topics, and follows. Ablation study on the filter items is detailed in the Appendix. E.2.

3. In the generation stage, we prompted actor nodes to act accordingly based on their observations. To this end, we define different action prompt templates based on the simulation scenario. The action prompt templates are defined in Table 19, Table 20, and Table 21.

Parallel Acceleration To enhance the simulation speed of GAG, we propose Nested-ACTOR based on the traditional actor architecture (Hewitt et al., 1973). As highlighted by (Clauset et al., 2004), network structures often exhibit densely connected communities with weaker inter-community links. To exploit this characteristic, the primary goal is to categorize agents into groups, each defined by an active actor agent paired with an item agent, thereby enabling parallel execution across these groups. We initialize a supervisor agent to manage the agents within each group, with each supervisor actor assigned to a different CPU core of the computational machine. Between groups, the item agent and active actor agent share a single message queue to facilitate intra-group message processing. Within groups, supervisor actors manage inter-group parallel message processing. Agents within a single group only need to account for the I/O wait times of other agents in that group, rather than waiting on all agents in the system.

A.3 Graph Projection

In various simulation scenarios, B_0 progressively evolves into B_K after K rounds of interaction simulations. As defined in Table 5, the bipartite graph is marked by nodes and edges of different types. Specifically, the action type marks the edge type; the item and actor type mark the node type; and the associated textual attribute marks the node attribute. The sub-graph, denoted as $G(\mathcal{V}^s, \mathcal{E}^s)$, is projected by B_K. Following established folding rules in network science research, the subgraphs embody different semantic interpretations.

In the context of SC, following the action template in Table 19, the author references other papers each time a paper is generated. To fold graphs from the pair-wise interaction process, we define the following mapping functions:

1. **Paper Citation:** Let $\mathcal{V}^s$ represents the set of papers, $\mathcal{E}^s$ represents the one paper is cited by another paper of one author, and $\mathcal{X}^{Vs}$ signify the textual attributes of each paper.

2. **Bib Coupling:** Let $\mathcal{V}^s$ represents the set of papers, $\mathcal{E}^s$ represents the relationships where two papers cite the same reference, and $\mathcal{X}^{Vs}$ encompasses the attributes of the papers.

3. **Co-citation:** Let $\mathcal{V}^s$ represents the set of papers, $\mathcal{E}^s$ represents the relationships where two papers are cited by the same paper, and $\mathcal{X}^{Vs}$ includes the attributes of the respective papers.

4. **Author Citation:** Let $\mathcal{V}^s$ represents the set of authors, $\mathcal{E}^s$ represents the papers of one author is cited by another author, and $\mathcal{X}^{Vs}$ refers to the attributes of each author.

5. **Co-Authorship:** Let $\mathcal{V}^s$ represents the set of authors, $\mathcal{E}^s$ represents the collaborative relationships between authors, and $\mathcal{X}^{Vs}$ characterizes the attributes of each author.

In the context of TC, the following action template in Table 20 is used each time the user generates a movie rating. To fold graphs from the pair-wise interaction process, we define the following mapping functions:

1. **Movie Rating**: Let $\mathcal{V}^s$ represents the movie watchers and the movies, $\mathcal{E}^s$ represents the movie ratings. For movie watchers, $\mathcal{X}^{Vs}$ corresponds to the attributes of movie watchers; for movies, $\mathcal{X}^{Vs}$ corresponds to the attributes of movies.

2. **User Projection**: Let $\mathcal{V}^s$ represent the movie watchers, $\mathcal{E}^s$ represent the movie watcher relationships who jointly rated movies, and $\mathcal{X}^{Vs}$ encompasses the attributes of the movie watchers.

In the context of SoC, the following action template in Table 21, each time the user generates a tweet and retweets/replies to other tweets and follows other users. To fold graphs from the pair-wise interaction process, we define the following mapping functions:

1. **Action:** Let $\mathcal{V}^s$ represent users, $\mathcal{E}^s$ denote the edges indicating tweets exchanged between two users (e.g., retweets, follow, reply actions), and $\mathcal{X}^{Vs}$ represent user attributes.

2. **Follow:** Let $\mathcal{V}^s$ represent users, $\mathcal{E}^s$ denote the edges indicating a follow relationship between two users, and $\mathcal{X}^{Vs}$ represent user attributes.

3. **Friend:** Let $\mathcal{V}^s$ represent users, $\mathcal{E}^s$ denote the edges indicating a friend relationship between two users (i.e., mutual following), and $\mathcal{X}^{Vs}$ represent user attributes.

B Graph Structure Alignment

For the LLM backbone, we have chosen the open-source model of Llama-3-70B(AI@Meta, 2024) for the large-scale graph generation in the macro-level structure alignment experiment. For micro-level structure alignment, we select the closed-source model of GPT-3.5-turbo for a more accurate simulation of human behaviors. Additionally, we select (Reimers and Gurevych, 2019) [2] as the encoder in S-RAG.

B.1 Graph Structure Metrics

To measure the structural characteristics of graphs, we use the following structural metrics:

(1) $|\mathcal{V}^s|$: measures the node number of graph $\mathcal{G}^s$.

(2) $|\mathcal{E}^s|$: measures the edge number of graph $\mathcal{G}^s$.

(3) $\bar{cc}$: average clustering coefficient, quantifies the degree to which nodes in a graph tend to cluster together [3].

[2] https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2

[3] https://en.wikipedia.org/wiki/Clustering_coefficient.

(4) r: assortativity, measures the similarity of connections in the graph concerning the node degree. [4]

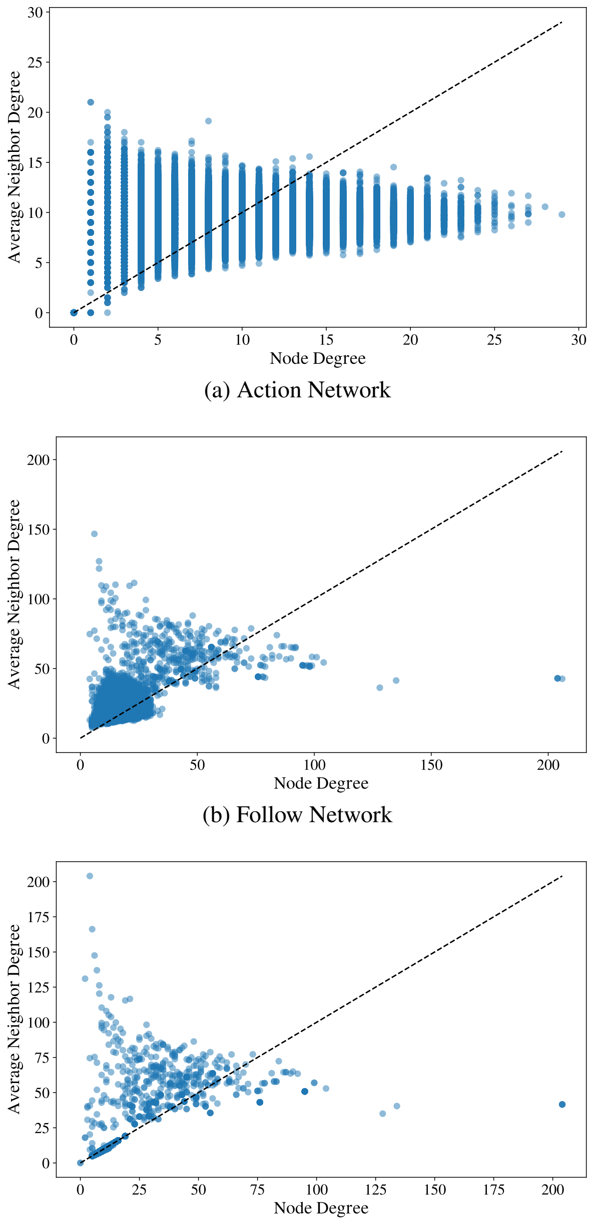
(5) D_e: effective diameter, defined as the minimum number of hops in which a certain percentage (typically 90% or 95%) of all connected node pairs can be reached.

As shown in Table 9, we calculate the structural metrics of all generated networks. Similar to the assortative-mixing patterns discovered in real-world networks (Newman, 2002), the citation network exhibits negative assortativity, whereas the co-authorship network displays positive assortativity. Moreover, the generated networks exhibit a small diameter, $D_e \in [1.17, 11.66]$, consistent with the *six degrees of separation* phenomenon observed in real-world networks (Leskovec et al., 2007; Broder et al., 2000). The high $\bar{cc}$, combined with small D_e, confirms these networks exhibit small-world characteristics.

B.2 Macro-Level Evaluation

Periodic Variation of Degree In the simulation scenario of TC, we filter the review data from the Movielens-25M dataset based on the movies listed in the Movielens-1M dataset. We select the top 10 ratings for each user and discover a noteworthy phenomenon: the number of reviews in the rating network exhibits periodic fluctuations over time. By scraping the release dates of the movies and plotting their release frequency, we observe that the periodicity in the release frequency is consistent with the fluctuations in the number of reviews. To quantify the periodicity, we selected the signal-to-noise ratio (SNR) (Johnson, 2006) as our metric, considering an SNR greater than 10 dB to indicate strong periodicity and reliability. Furthermore, we observe that the degree of the generated rating graph also exhibits periodic variations consistent with the release dates of the movies. As illustrated in Figure 4, the SNR of the degree of the rating graph is 12.79 dB, surpassing the 10 dB threshold, thus demonstrating significant periodic fluctuations.

Emergent of GCC In online social graphs, nodes with higher degrees grow larger over time and eventually manifest a giant connected component (GCC) (Mislove et al., 2007). As illustrated in Figure 5, the proportion of the largest connected component rows steadily increases over time, indicating the emergence of a giant community within the social graph. The network is generated with 7000 actor agents.



(c) Friend Network

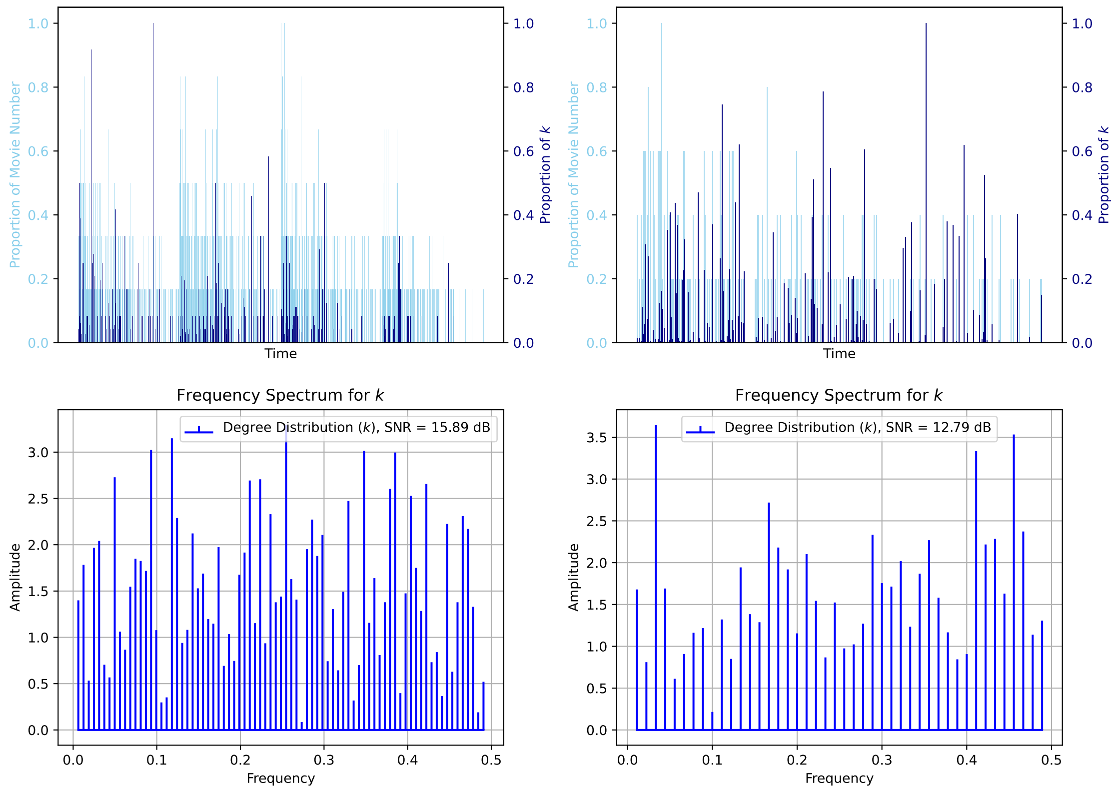
Figure 6: The *Friendship Paradox* phenomenon in online social graphs; The figures show the average degree of node neighbors v.s. the average degree of the node itself in social graphs.

Friendship Paradox An interesting and somewhat counterintuitive phenomenon in real-world social graphs is that everyone you follow or who follows you tends to have more friends and followers than you do. This phenomenon has been observed in both Twitter (Hodas et al., 2013) and

[4] https://en.wikipedia.org/wiki/Assortativity

Table 9: The structural metrics for graphs generated by GAG.

	Citation	Bib-Coupling	Co-Citation	Author Citation	Co-Authorship
$\|\mathcal{V}^s\|$	1.14e+04	1.09e+04	3.93e+03	5.01e+03	5.01e+03
$\|\mathcal{E}^s\|$	3.63e+04	1.22e+07	3.27e+04	2.41e+05	2.08e+04
$\bar{cc}$	0.08	0.77	0.59	0.38	0.20
r	-0.10	0.09	-0.10	-0.18	0.32
D_e	5.19	2.94	3.89	3.44	5.77
	Action	Follow	Friend	Movie Rating	User Projection
$\|\mathcal{V}^s\|$	9.97e+04	9.96e+04	9.96e+04	4.17e+03	3.91e+03
$\|\mathcal{E}^s\|$	9.07e+05	1.53e+06	5.01e+05	3.25e+04	9.04e+05
$\bar{cc}$	0.07	0.61	1.00	0.00	0.34
r	-0.03	0.06	0.59	-0.54	-0.11
D_e	2.79	2.85	11.66	2.98	1.17



(a) Movielens Rating Network

(b) Generated Movie Rating Network

Figure 4: Periodic Variation of Degree in Movie Rating Network; Figure 4a shows the number of released movies over time and the degree of the movie rating network over time in MovieLens dataset; Figure 4b also shows the number of released movies and the degree of the movie rating network over time in GAG.

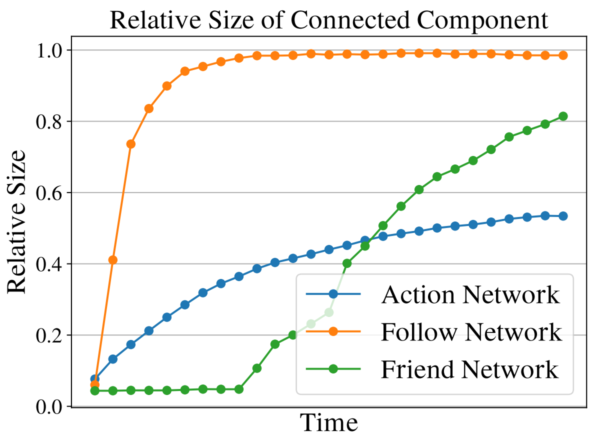


Figure 5: The proportion of the largest connected component grows steadily over time.

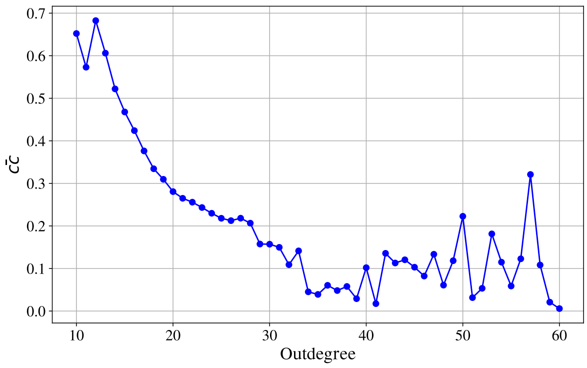


Figure 8: The D_e of the DCC in the follow network, the network is divided by the fractions of the total number of nodes.

the social graph of Facebook (Ugander et al., 2011), applying to more than 98% of the nodes. As shown in Figure 6, the friendship paradox is most evident in the friend network, with over 90% of the nodes lying above the $y = x$ line, indicating that most users have fewer friends than their friends do. The network is generated by five rounds of simulation with 1e5 agents.

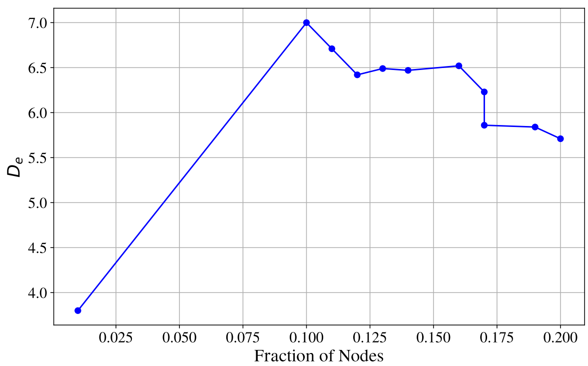


Figure 7: The out-degree is plotted against $\bar{cc}$ in follow graph.

Densely Connected Core In real-world online social graphs, there exists a densely connected core (DCC) comprising between 1% and 10% of the highest degree nodes, such that removing this core completely disconnects the graph (Mislove et al., 2007). These high-degree nodes serve as hubs of the network, causing the network to become increasingly compact through the hub structure. As shown in Figure 8, the nodes with higher degrees have significantly higher $\bar{cc}$ compared to other nodes. For these densely connected components, D_e grows at a slow rate. In Figure 7, we observe that the D_e among the DCC of the follow network grows sublogarithmically. The network is generated by five rounds of simulation with 1e5 agents.

B.3 Micro-Level Evaluation

GAG distinguishes itself from traditional graph generation methods by generating graph data without requiring prior training. It achieves this through the simulation of human behavior, leading to the emergence of various structural features characteristic of real-world networks. Consequently, conventional graph evaluation methods cannot be applied. To address this, we design comparative experiments against existing graph generation models.

Evaluation Metrics In accordance with established evaluation metrics for graph generation (Bergmeister et al., 2024), we report the maximum mean discrepancy (MMD) between the generated graphs and the test graphs, specifically focusing on degree distribution and clustering coefficient. Furthermore, we place particular emphasis on evaluating whether the degree distribution of the generated graphs conforms to a power law after expanding the graph to out-of-distribution sizes (Clauset et al., 2009). To this end, we employ six key metrics in all:

(1) MMD.D: maximum mean discrepancy (MMD) of degree distribution between the generated graphs and the test graphs.

(2) MMD.C: maximum mean discrepancy (MMD) of clustering coefficient between the generated graphs and the test graphs.

(3) MMD.S: maximum mean discrepancy (MMD) of spectrum between the generated graphs and the test graphs (Liao et al., 2019).

(4) MMD.O: maximum mean discrepancy

Table 10: Training hyperparameters of baseline models. All unspecified hyperparameters default to their standard values.

Model	Hyperparameter	Experiment
Erdös-Rényi (Erdos et al., 1960)	Linking propoblity	$\bar{k}/(\lvert\mathcal{V}^s\rvert - 1)$
Barabási-Albert (Barabási and Albert, 1999)	Number of linking edges	$\bar{k}/2$
Small-World (Watts and Strogatz, 1998)	Number of linking nodes	$\bar{k} \times 2$
SBM (Holland et al., 1983)	Block sizes	$\lvert\mathcal{V}'\rvert//6$
	p	0.1
GN (Krapivsky and Redner, 2001)	Number of generated nodes	$\lvert\mathcal{V}'\rvert$
BiGG (Dai et al., 2020)	Ordering	DFS
	Accumulated gradients	1
	Batch size	32
GRAN (Liao et al., 2019)	Hidden size	512
	Embedding size	512
	Number of layers	7
	Number of mixtures	20
	Batch size	16
BwR (Diamant et al., 2023)	Model	GraphRNN (You et al., 2018)
	bw	8
	Hidden size	128
	Ordering	BFS
	Batch size	32
L-PPGN (Bergmeister et al., 2024)	Hidden embedding size	256
	PPGN embedding size	128
	Input embedding size	32
	Number of layers	10
	Number of denoising steps	1024
	Batch size	32
	EMA coefficient	0.99
	Number of spectral features	0
GraphMaker (Li et al., 2024a)	Variant	Sync
	Hidden size for timestep	32
	Hidden size for node	512
	Hidden size for node label	64
	NumberofMPNNlayers	2
	Learning rate	0.001
	Optimizer	AMSGrad

Table 11: Comparison with existing expansion-based graph generation models. For GRAN, the generated graph degree distribution fails to converge when fitting a power-law distribution.

	MMD.D↓	MMD.C↓	MMD.S↓	MMD.O↓	D_k	α	Valid↑	GEM
Cora	-	-	-	-	$0.07_{\pm 0.0}$	$2.59_{\pm 0.01}$	1.00	-
Erdös-Rényi	0.25	1.41	0.54	0.27	$0.13_{\pm 0.02}$	$4.01_{\pm 0.17}$	0.00	0.29
Barabási-Albert	**0.09**	1.41	0.44	1.11	$0.04_{\pm 0.01}$	$2.4_{\pm 0.05}$	**1.00**	0.46
Small-World	0.60	1.41	0.50	<u>0.20</u>	$0.14_{\pm 0.0}$	$4.05_{\pm 0.02}$	0.00	0.28
SBM	0.88	1.17	0.83	1.02	$0.60_{\pm 0.00}$	$1.24_{\pm 0.00}$	0.00	0.22
GN	<u>0.10</u>	1.41	0.45	1.13	$0.04_{\pm 0.01}$	$2.38_{\pm 0.03}$	**1.00**	0.46
BiGG	0.14	0.51	0.48	0.27	$0.08_{\pm 0.01}$	$3.19_{\pm 0.11}$	0.05	0.34
GRAN	0.15	<u>0.50</u>	0.55	0.28	-	-	0.35	0.42
BwR	0.32	**0.23**	<u>0.34</u>	**0.19**	$0.1_{\pm 0.01}$	$3.58_{\pm 0.08}$	0.00	0.35
GraphMaker	0.37	1.41	0.75	0.28	-	-	0.00	0.27
L-PPGN	0.15	0.92	**0.32**	0.59	$0.06_{\pm 0.01}$	$2.77_{\pm 0.04}$	**1.00**	0.50
GAG	0.35	0.84	0.41	1.21	$0.09_{\pm 0.0}$	$2.1_{\pm 0.01}$	**1.00**	0.47

(MMD) of node orbit counts between the generated graphs and the test graphs (You et al., 2018).

(5) α**:** The power-law exponent of the graph degree distribution.

(6) D_k**:** The Kolmogorov-Smirnov distance between the degree distributions of the generated and test graphs.

(7) Valid: Research demonstrates that degree distributions in complex networks are typically characterized by a power-law exponent $\alpha \in [2, 3]$ (Clauset et al., 2009). Accordingly, we define the validity measure for a graph as the proportion of graphs meeting the criteria $D_k < 0.1$ and $\alpha \in [2, 3]$. Set $k_{\text{min}} = 2$ for the uniform calculation of the power-law fitness of both undirected and directed graphs.

(8) GEM: To quantify the level of structural alignment for the expanded graph, we establish the Graph Expansion Metric (GEM). Firstly, for the negative indicator MMD metrics, we utilize the transformation $1 - \frac{1}{1+e^{metric}}$, which maps the metrics to a range between 0 and 1. We then calculate the average of MMD and Valid metrics as GEM.

Experiment Settings Specifically, we sample a network dataset based on publication timelines to create our evaluation dataset. Since we only crawl for timestamp information of the CiteSeer and Cora datasets, these two datasets are used for our experimental evaluation. Following the timeline of graph evolution, we partition the network dataset into training and testing sets. At a designated time point t, we filter the citation network using node timestamp information to obtain $G_{<t}$, which includes all nodes and edges prior to t. We sample small subgraphs from $G_{<t}$ to create a training set for deep learning methods and to generate the seed graph for GAG. The training set consists of sampled subgraphs with sizes ranging from 64 to 512 nodes, resulting in a training set comprising 160 subgraphs and a validation set comprising 32 subgraphs. For the test set, we filter the citation network for nodes and edges after t, denoted as $G_{>t}$. From $G_{>t}$, we sample large subgraphs of 1,000 nodes, resulting in a test set comprising 20 subgraphs. We focus on whether the expanded graph structure exhibits power law characteristics typical of real-world network structures.

The existing graph generation methods have demonstrated promising results in generating small graphs, including works such as (Vignac et al., 2023), (Martinkus et al., 2022), and (You et al., 2018). However, they have not explored the generation of out-of-distribution graph sizes, and the generated graph sizes are limited. To compare with traditional graph generation models, we need to select those methods that can expand beyond the training set graph sizes and efficiently generate large graphs. For rule-based graph generation methods, we set hyperparameters to ensure that the average degree of the expanded graph matches that of the seed graph. For deep learning-based graph generation methods, we adhere to the hyperparameters specified in the original papers. All hyperparameter details are provided in Table 10.

Ablation on Seed Graph Size Additionally, we aim to investigate whether the size of the seed graph affects the validity of the final generated graph structure. To this end, we utilize the GAG to perform graph expansion on seed graphs of varying sizes. We plot the number of nodes in the expanded

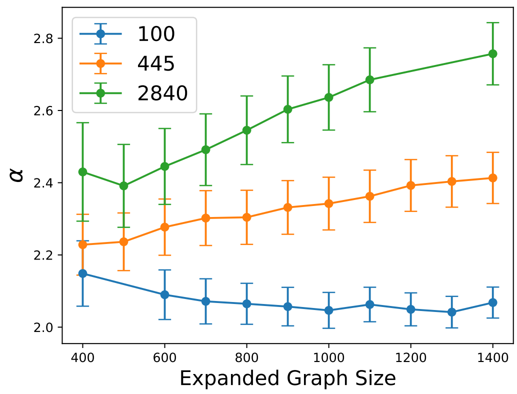


Figure 9: We present the results for seed graph sizes of 100, 445, and 2840, with the sizes of the expanded graphs plotted against their corresponding α values. Note that we only plot valid data $D_k < 0.1$.

graphs against the corresponding values of α. As shown in Figure 9, it is evident that larger seed graphs result in expanded graphs exhibiting higher values of α. Furthermore, as the size of the expanded graphs increases, the α values gradually stabilize. This indicates that the GAG is capable of effectively and reasonably expanding graphs across different seed graph sizes.

Supplementary Experiments To further demonstrate the reliability of the GAG framework, in addition to seed graphs from CiteSeer, we crawl for the necessary node attributes to enrich text attributes for Cora (Sen et al., 2008) to constitute the seed graph. Following the experimental setup outlined in the paper, we designate Cora as the seed graph and similarly compare it with existing graph generation models.

As shown in Table 11, the expanded graphs generated by GAG adhere to a power-law distribution, with $\alpha = 2.1$. For the MMD metrics, since GAG does not strictly enforce explicit graph structure constraints on the agents, its performance is not significantly superior to that of other models. However, it achieves comparable results. In the case of the Cora dataset, the seed graphs are more densely connected compared to CiteSeer, which makes it easier for deep learning-based models to learn the underlying graph structure from the seed graph training set. As a result, baseline models perform better in terms of MMD on the Cora dataset. For the Valid metric, apart from GAG and the Barabási-Albert Model, the only existing deep-learning graph generation model capable of capturing the power-law distribution is L-PPGN. This demonstrates that L-PPGN is indeed capable of extrapolating to out-of-distribution graphs when trained on graphs that encompass the power-law distribution property. However, the unstable performance of L-PPGN across different datasets also highlights its sensitivity to the quality of the training dataset. In contrast, GAG, through human behavior simulation, can reliably generate graph structures that adhere to real-world network characteristics from a seed graph of varying sizes and quality. This illustrates not only the potential of LLM-based Agents in simulating human behavior but also underscores the reliability of the GAG framework.

C Textual Feature Alignment

C.1 Experiment Settings

For the LLM backbone, we have chosen the open-source model of Llama-3-70B(AI@Meta, 2024) for textual feature alignment experiment. We configure the complete Citeseer graph as seed graph, and expand the paper citation graph to 5000 nodes. For baselines, we construct graphs with eight distinct relationships between graph structures and textual features as follows:

(1) F.shuffle: This process randomly selects node-wise textual features from seed graph to create the node features of expanded graph.

(2) S.shuffle: In this step, we randomly shuffle the edges of seed graph, thereby disrupting the graph structure while maintaining the number of edges consistent with $\mathcal{V}^s$.

(3) SF.shuffle: This refers to a combination of both F.shuffle and S.shuffle.

(4) BWR.L: Graph strcuture generated by (Diamant et al., 2023). We give the LLM Citeseer as corpus and use it to generate textual features.

(5) L-PPGN.L: Graph strcuture generated by (Bergmeister et al., 2024). We give the LLM Citeseer as corpus and use it to generate textual features.

(6) BiGG.L: Graph strcuture generated by (Dai et al., 2020). We give the LLM Citeseer as corpus and use it to generate textual features.

(7) BiGG.L: Graph strcuture generated by (Dai et al., 2020). We give the LLM Citeseer as corpus and use it to generate textual features.

(8) GraphMaker.L: Graph strcuture generated by (Li et al., 2024a). We give the LLM Citeseer as corpus and use it to generate textual features.

(9) GAG: Graphs that are generated by the GAG framework.

C.2 Ablation Study of LLM

To further illustrate the effectiveness of the generated graphs, we conduct an ablation study on the LLM for agent setup. We select four LLMs for this study: GPT-3.5-turbo, GPT-4o-mini (OpenAI, 2023) as top-ranking closed-source LLMs, and Llama-3-70B(AI@Meta, 2024) and Qwen2-72B(Yang et al., 2024) as top-ranking open-source LLMs.

As shown in Table 12, the LLM-based agents built on LLaMA2, GPT-3.5-turbo, and GPT-4o-mini are capable of generating graphs that maintain the node and structural characteristics of the seed graph, thus ensuring effective performance transfer in downstream tasks. In contrast, Qwen-2 based agents do not guarantee performance transfer. We believe this is related to the ability of LLMs to emulate human behavior; Qwen-2 based agents fail to exhibit human-like creative behavior, resulting in less coherent generated graphs.

D Scalability of GAG

GAG demonstrates the ability to generate text-attributed dynamic graphs at scales exceeding those of existing graph generation models. Most current methods are limited to producing graphs with up to 5,000 nodes, while specialized models designed for sparse large graphs, such as Dai et al. (2020), are constrained to generating simple grid-structured graphs with a maximum of 100,000 nodes. Similarly, the GraphMaker model (Li et al., 2024a) considers graphs with 13,000 nodes to be large-scale. In contrast, our GAG framework is capable of generating graphs with up to 100,000 nodes that exhibit intricate small-world structures, without imposing sparsity assumptions. Moreover, GAG produces graphs that are both dynamic and text-attributed—a substantial advancement over existing approaches. While some models can generate either dynamic or attributed graphs, none match GAG's ability to support the generation of text-attributed dynamic graphs at such scales, marking a key contribution of our work. To further clarify the scalability of the GAG framework, we have included an updated comparison in Table 13:

Table 14: The time cost (*min*) of agents for generating one interaction data with $P = 24$.

N	SC	TC	SoC
5	0.2700	0.0150	0.0120
10	0.2300	0.0112	0.0119
20	0.1700	0.0052	0.0119
40	0.0910	0.0054	0.0060
5→40	↓**66.3%**	↓**64.0%**	↓**50.0%**

Table 15: The time cost (*hour*) for one round of simulation for generating the large-scale graphs.

	N	P	T
SC	5.01e+03	10	0.46h
TC	3.91e+03	10	0.30h
SoC	9.97e+04	48	11h

To further demonstrate the excellent scalability of the GAG framework, we conduct time measurements across various simulation scenarios. The tests are carried out on a computing machine equipped with 96 CPU cores and 376 GB of memory. For model inference, we utilized LLAMA-3-70B as the backbone LLM and employed the vLLM framework (Kwon et al., 2023), running on a setup of four A-800 GPUs. As shown in Table 14, when P is held constant, the time to generate one interaction data decreases as N increases. The most significant time reduction observed in the SC simulation is where agents are grouped by paper authorship, which maximizes the efficiency of parallel acceleration.

For generating the large-scale graphs listed in Table 9, we carry out SC simulation experiments for 200 rounds, TC simulation experiments for 33 rounds, and SoC simulation experiments for 10 rounds. For each scenario, we measured the total simulation time and computed the average simulation time per round. These multi-round simulations provide a robust measure of the computational efficiency of the GAG framework. The results, summarized in Table 15, demonstrate GAG's capability to handle simulations with varying scales of agents. For simulation experiments with Thousands of agents, the average simulation time per round is 0.46 hours for SC experiments and 0.30 hours for TC experiments; For simulation experiments with Hundreds of thousands of agents, the average simulation time per round is 11 hours for

Table 12: Ablation Study of LLM in GAG. Performance Comparison for Node Classification

		$\Delta ACC \downarrow$			
GNN	LLM	SF.random	F.random	S.random	GAG
GAT	LLAMA.	$13.40_{\pm 3.52}$	$24.11_{\pm 2.84}$	$18.85_{\pm 2.55}$	$\mathbf{2.26}_{\pm 1.19}$
	GPT-3.	$12.30_{\pm 2.47}$	$22.24_{\pm 2.86}$	$16.69_{\pm 2.46}$	$\mathbf{2.29}_{\pm 1.84}$
	GPT-4o.	$12.03_{\pm 1.17}$	$21.73_{\pm 2.91}$	$15.57_{\pm 2.84}$	$\mathbf{2.80}_{\pm 0.88}$
	Qwen2.	$11.12_{\pm 2.53}$	$15.84_{\pm 3.34}$	$24.51_{\pm 2.28}$	$\mathbf{10.50}_{\pm 1.94}$
GCN	LLAMA.	$15.04_{\pm 1.99}$	$23.20_{\pm 2.25}$	$21.77_{\pm 2.04}$	$\mathbf{3.61}_{\pm 1.32}$
	GPT-3.	$13.55_{\pm 2.19}$	$23.08_{\pm 1.39}$	$20.58_{\pm 1.44}$	$\mathbf{4.34}_{\pm 2.01}$
	GPT-4o.	$14.56_{\pm 1.84}$	$22.07_{\pm 2.25}$	$20.82_{\pm 2.45}$	$\mathbf{4.09}_{\pm 1.75}$
	Qwen2.	$12.56_{\pm 1.65}$	$15.37_{\pm 2.69}$	$26.95_{\pm 2.16}$	$\mathbf{10.05}_{\pm 1.87}$
GCN2	LLAMA.	$9.58_{\pm 0.88}$	$9.33_{\pm 1.36}$	$2.24_{\pm 1.58}$	$\mathbf{0.49}_{\pm 1.50}$
	GPT-3.	$9.07_{\pm 1.10}$	$9.04_{\pm 1.80}$	$1.43_{\pm 1.67}$	$\mathbf{0.39}_{\pm 1.67}$
	GPT-4o.	$8.89_{\pm 1.44}$	$9.64_{\pm 1.10}$	$1.19_{\pm 1.64}$	$\mathbf{0.50}_{\pm 1.36}$
	Qwen2.	$9.78_{\pm 1.48}$	$\mathbf{9.19}_{\pm 1.44}$	$11.40_{\pm 1.78}$	$9.39_{\pm 0.93}$
GraphSage	LLAMA.	$7.00_{\pm 0.92}$	$7.13_{\pm 1.73}$	$3.15_{\pm 1.84}$	$\mathbf{0.09}_{\pm 1.74}$
	GPT-3.	$6.72_{\pm 1.41}$	$6.39_{\pm 0.92}$	$2.35_{\pm 1.60}$	$\mathbf{0.81}_{\pm 1.67}$
	GPT-4o.	$6.97_{\pm 0.92}$	$6.46_{\pm 1.61}$	$2.05_{\pm 2.31}$	$\mathbf{1.70}_{\pm 2.28}$
	Qwen2.	$\mathbf{6.45}_{\pm 1.33}$	$7.14_{\pm 1.68}$	$12.73_{\pm 2.26}$	$9.72_{\pm 2.43}$

SoC.

E Ablation Study of S-RAG

To investigate whether the hyperparameter settings of S-RAG affect the generated network structure. We conduct ablation experiments on these hyperparameters. Given the variations in graph generation scenarios, we conduct the ablation experiments under the SoC simulation. We run an equal number of simulation rounds within the GAG to generate graphs.

In this section, we add a graph structure metric for measuring the proportion of the largest connected component within the network. We define the largest connected component of the graph as LCC, so the proportion of LCC within the network is $|LCC|/|V|$. This aids us in comprehending the graph evolution progress.

E.1 Recall Stage

In the recall stage, the only hyperparameter is the number of searched items: N_r. Since the final number of documents interacting with the LLM is limited to N_r, we change N_r and evaluate its impact on network structure.

As shown in Table 16, we keep all other search parameters constant while varying the size of N_r. It can be observed that as N_r increases, $\bar{k}$ of generated network also exhibits an upward trend. This trend is particularly pronounced in the follow network.

E.2 Reranking Stage

To maximum the effectiveness of searched items, we implement the ReRanking stage in S-RAG. Initially, coarse ranking is performed to sort the searched items by their creator agent. Focusing on the *core* label of creator. Subsequently, fine ranking is conducted based on the agent's individual preferences. We conduct an ablation study to explore the impact of different levels of personalization in the ReRanking stage. And eventually, its impact on the network structure.

We focus on the hyperparameters in the ReRanking stage, which mainly include: (1) Hub rate: $|HUB|/|V|$. (2) Attributes of a_l.

Coarse Ranking As shown in Table 17, an increase in the proportion of core users correlates with an upward trend in the proportion of the largest connected component within the network. Since the proportion of core users is increased, the likelihood of core users being searchable by general users is also increased, thereby fostering preferential attachment in the network. Eventually, the proportion of the largest connected component within the network increases.

Table 13: Comparsion of generated graphs by GAG against existing graph generation methods.

Model	Method	Text-Attributed	Attributed	Temporal	Scale(Nodes)	Year
GAG	Simulation	✓	✓	✓	100000	2024
GraphMaker (Li et al., 2024a)	Diffusion		✓		13000	2024
L-PPGN (Bergmeister et al., 2024)	Diffusion		✓		5037	2024
EDP-GNN (Niu et al., 2020)	Diffusion		✓		<100	2020
MOOD (Lee et al., 2023)	Diffusion		✓		<100	2022
GDSS (Jo et al., 2022)	Diffusion		✓		<100	2022
DiGress (Vignac et al.)	Diffusion		✓		<100	2021
Bwr-GraphRNN (Diamant et al., 2023)	AR				5037	2023
BIGG (Dai et al., 2020)	AR				100000	2020
GRAN (Liao et al., 2019)	AR				5037	2019
GraphRNN (You et al., 2018)	AR				2025	2018
MolecularRNN (Popova et al., 2019)	AR		✓		<100	2019
GRAM (Kawai et al., 2019)	AR				500	2021
DeepGDL (Khodayar et al., 2019)	AR				14430	2019
LFM (Podda et al., 2020)	AR				<100	2020
STGG (Ahn et al., 2021)	AR				<100	2021
MDVAE (Du et al., 2022b)	VAE		✓		<100	2022
D-MolVAE (Du et al., 2022c)	VAE				<100	2022
GraphVAE (Simonovsky and Komodakis, 2018)	VAE		✓		<100	2018
STGD-VAE (Du et al., 2022a)	VAE		✓	✓	2500	2022
CGVAE (Liu et al., 2018)	VAE		✓		<100	2018
JT-VAE (Jin et al., 2018)	VAE		✓		<100	2018
GraphNVP (Madhawa et al., 2019)	NF		✓		<100	2019
GRF (Honda et al., 2019)	NF		✓		<100	2019
MoFlow (Zang and Wang, 2020)	NF		✓		<100	2020
GraphAF (Shi et al., 2020)	AR+NF		✓		<100	2019
GraphDF (Luo et al., 2021)	NF		✓		<100	2021
MolGAN (De Cao and Kipf, 2018)	GAN		✓		<100	2018
Mol-CycleGAN (Maziarka et al., 2020)	GAN		✓		<100	2020
GraphEBM(Liu et al., 2021)	EBM		✓		<100	2021

Table 16: Ablation Study of N_r. The value of N_r is proportional to $\bar{k}$ of the generated network.

Network	N_r	$\lvert\mathcal{V}^s\rvert$	$\lvert\mathcal{E}\rvert$	$\bar{cc}$	r	$\lvert LCC\rvert/\lvert V\rvert$
Action	3	9.47e+02	1.92e+03	0.07	0.10	0.03
	5	9.36e+02	2.20e+03	0.09	-0.05	0.02
	10	9.58e+02	2.63e+03	0.09	0.02	0.05
	20	1.03e+03	3.03e+03	0.11	-0.08	0.16
Follow	3	7.42e+02	1.27e+04	0.83	-0.08	0.29
	5	7.39e+02	1.24e+04	0.81	-0.06	0.44
	10	7.39e+02	1.29e+04	0.80	-0.06	0.51
	20	8.91e+02	3.83e+04	0.82	-0.18	1.00
Friend	3	7.42e+02	5.96e+03	0.88	-0.13	0.25
	5	7.39e+02	5.76e+03	0.87	-0.10	0.22
	10	7.39e+02	5.94e+03	0.89	-0.10	0.24
	20	8.91e+02	1.83e+04	0.87	-0.10	0.45

Table 17: Ablation Study of the hub rate ($\lvert HUB\rvert/\lvert V\rvert$). Higher hub rate contributes to the emergence of a large connected component.

Network	$\lvert HUB\rvert/\lvert V\rvert$	$\lvert\mathcal{V}^s\rvert$	$\lvert\mathcal{E}^s\rvert$	$\bar{cc}$	r	$\lvert LCC\rvert/\lvert V\rvert$
Action	0.00	9.78e+02	2.64e+03	0.09	-0.05	0.13
	0.10	1.02e+03	2.58e+03	0.09	-0.07	0.10
	0.20	1.03e+03	3.03e+03	0.11	-0.08	0.16
Follow	0.00	7.79e+02	3.00e+04	0.84	0.02	0.63
	0.10	7.82e+02	3.04e+04	0.84	0.05	0.63
	0.20	8.91e+02	3.83e+04	0.82	-0.18	1.00
Friend	0.00	7.79e+02	1.45e+04	0.89	0.21	0.27
	0.10	7.82e+02	1.47e+04	0.88	0.29	0.43
	0.20	8.91e+02	1.83e+04	0.87	-0.10	0.45

Fine Ranking To improve search algorithms based on personal preferences of agents, we design various filter items in the fine ranking process, which are tailored to different simulation scenarios. The number of filter items is N_f. Within the SoC simulation, filter items include: (1) Follow: Assesses whether the content of the document is posted by an agent that the current agent follows. (2) Friend: Assesses whether the content of the document is sent by an agent that is a friend of the current agent. (3) Topic: Assesses whether the content of the document is related to a topic that the current agent is interested in.

As illustrated in Table 18, $\bar{cc}$ of network increases as N_f increases. Additionally, the impact level of different filter items is as follows: friend > topic > follow.

F Human Interface Control

Previous work on employing LLMs for graph generation typically relied on predefined network structure features or a set of example networks (Yao et al., 2024a). Similarly, after understanding the reasons behind different structural characteristics of networks within GAG, we aim to enable users to control the entire simulation process by inputting prompts. This will guide and influence the various structural features of the final network.

To achieve this, we establish a control agent that

Table 18: Ablation Study of the filter items used in the fine ranking process.

Network	Fillter Items			Network Structural Characteristics				
	follow	topic	friend	$\|\mathcal{V}^s\|$	$\|\mathcal{E}^s\|$	$\bar{cc}$	r	$\|LCC\|/\|V\|$
Action	✓	-	-	6.51e+02	1.88e+03	0.11	0.01	0.19
	-	✓	-	6.32e+02	1.82e+03	0.09	-0.01	0.15
	-	-	✓	1.02e+03	2.78e+03	0.09	-0.05	0.12
	✓	✓	✓	1.03e+03	3.03e+03	0.11	-0.08	0.16
Follow	✓	-	-	6.51e+02	2.65e+04	0.78	-0.19	0.92
	-	✓	-	5.63e+02	2.03e+04	0.79	0.16	0.64
	-	-	✓	7.70e+02	2.97e+04	0.84	0.19	0.69
	✓	✓	✓	8.91e+02	3.83e+04	0.82	-0.18	1.00
Friend	✓	-	-	6.51e+02	1.26e+04	0.83	-0.13	0.46
	-	✓	-	5.63e+02	9.74e+03	0.83	0.52	0.28
	-	-	✓	7.70e+02	1.44e+04	0.89	0.21	0.31
	✓	✓	✓	8.91e+02	1.83e+04	0.87	-0.10	0.45

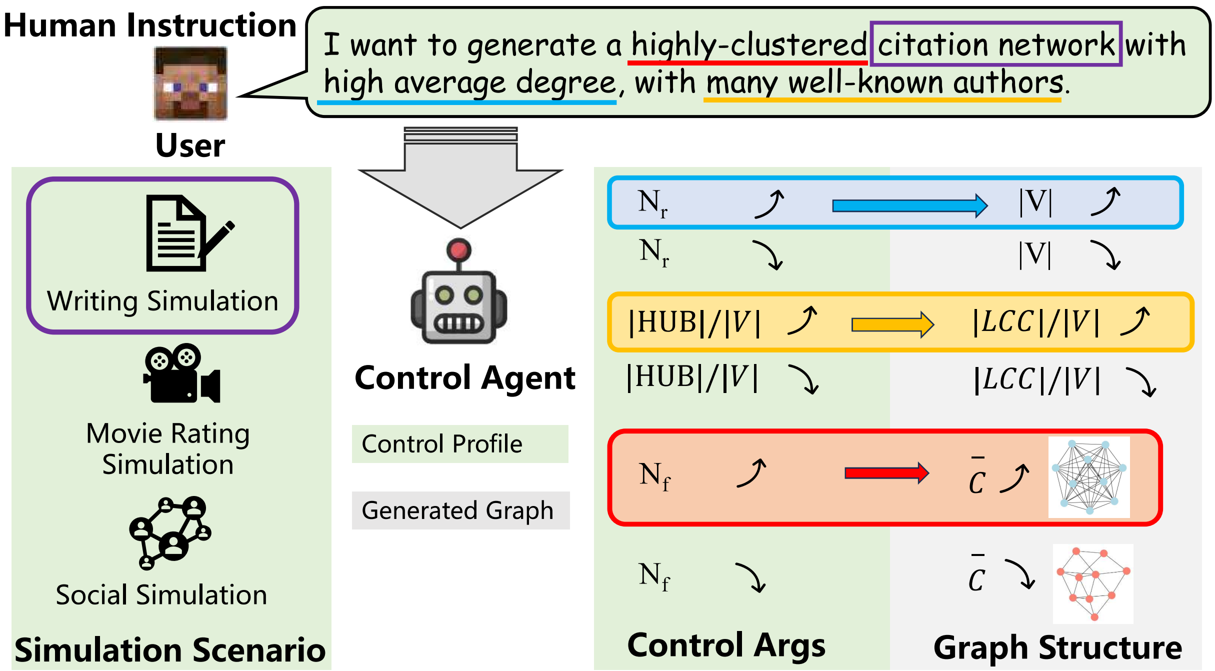


Figure 10: An illustration of the Control Agent in GAG Framework.

accepts instruction frm users. Control agent transfers the instruction to a control profile for managing the simulation process of GAG. As shown in Fig. 10, specifically, the hub rate controls the proportion of recommended core users, subsequently affecting the ratio of hub nodes in the network. The parameter N_r determines the number of items recommended by the system, influencing the overall degree distribution. Additionally, parameter N_f dictates the number of filter items in the ReRanking stage, impacting the network's clustering coefficient. Furthermore, the overall simulation time is adjusted by the number of agents N per simulation round.

F.1 Case Study

Since GAG employs human behavior simulation for network generation, the process of connecting each network node to others closely mirrors real-world scenarios. This alignment enables a clear and interpretable understanding of the network evolution process. To demonstrate the interpretability of our graph generation method, we present a case study using the SC simulation scenario.

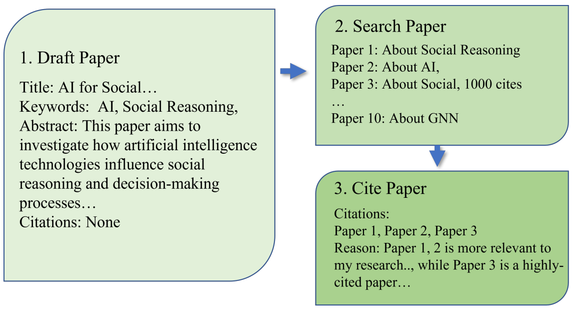


Figure 11: An illustration of the citation network evolution with LLM-based Agents.

As illustrated in Figure 11, the formation of a citation network involves three primary steps. First, LLM-based agents collaboratively generate a paper draft through interaction and cooperation. Next, the agents search the corpus of stored papers within the environment to identify literature relevant to their research interests. For instance, query terms may correspond to research domains such as AI or social sciences.

Finally, after completing the search, the agents select and cite papers pertinent to their draft, providing explicit justifications for each citation. As illustrated in Figure 11, examples of such justifications include citing a paper due to its high citation count or its direct relevance to the research topic. Each citation edge in the network thus directly corresponds to an agent's citation action, offering a behavior-driven perspective on the graph construction process. This approach ensures that the graph generation process is inherently interpretable.

Table 19: The action prompt template and respond example in SC.

Action Prompt Template
Agent Profile: <Agent Profile>
Agent Memory: <Agent Memory>
Environment Observation: Partial environmental feedback provided by S-RAG. The searched items are formed as textual representation with item prompt template.
Human Instruction: A papar should include the following attributes:
title: The title should be concise yet descriptive, providing a clear indication of the paper's topic and scope. This can be different from your topic, It is relatively accurate and clear.
keywords: These are specific terms or phrases that encapsulate the core topics of your paper. Keywords make your paper searchable within academic databases and help readers quickly understand the paper's focus areas.
abstract: The abstract is a brief summary of your research paper. It should provide an overview of the research question, methodology, results, and conclusions.
citations: A list of the paper names you want to cite.
...
Now write a version of your paper and cite the papers you need to cite.

Structured Respond Example
Creation Action: Actor-Item Edge: Creation. Item Node: NULL.
Reference Action: Actor-Item Edge: Reference. Item Node: Academic paper.

Table 20: The action prompt template and respond example in TC.

Action Prompt Template
Agent Profile: <Agent Profile>
Agent Memory: <Agent Memory>
Environment Observation: Partial environmental feedback provided by S-RAG. The searched items are formed as textual representation with item prompt template.
Human Instruction: You should give your rating scores to the movies ...

Structured Respond Example
Movie Rating Action: Actor-Item Edge: Rating. Item Node: NULL.

Table 21: The action prompt template and respond example in SoC.

Action Prompt Template
Agent Profile: <Agent Profile>
Agent Memory: <Agent Memory>
Environment Observation: Partial environmental feedback provided by S-RAG. The searched items are formed as textual representation with item prompt template.
Human Instruction:
You can perform [Retweet/Reply/Tweet] action on these tweets. Additionally, you can follow the bloggers of these tweets:
Retweet: Retweet the tweet
Reply: Reply to the tweet
Tweet: Send a tweet
...

Structured Respond Example
Retweet Action: Actor-Item Edge: Retweet. Item Node: NULL.
Reply Action: Actor-Item Edge: Reply. Item Node: NULL.
Follow Action: Actor-Item Edge: Follow. Item Node: NULL.
Creation Action: Actor-Item Edge: Tweet. Item Node: Tweets.

Table 22: The profile prompt template in SC.

I would like you to generate a series of random author's personal information.
These authors are interested in computer science, they are experts in various fields of CS.
I need you to give a list of author infos with the constraints for each attribute as follows:
(1) Name: Author's name
(2) Expertises: a list, The author's areas of expertises can be selected from the following areas:{expertises list}
(3) Institution: The author's institution, you can choose whatever institution you want, just give me one institution name
(4) Country: The author's country, you can choose whatever institution you want,just give me one country name corresponding to the institution
(5) Topics: a list, The topics this author is interested in, can be selected from the following topics:{topics list}
Here's some common used countrys you can infer to:
{countrys}
Please generate me a list of {author num} different authors, which can be loaded by eval function in python:
[{{
"name":"",
"expertises":[],
"institution":"",
"country":"",
"topics":[]
}},
...,
{{
"name":"",
"expertises":[],
"institution":"",
"country":"",
"topics":[]
}}]
Now please generate:

Table 23: The profile prompt template in TC.

Your task is to give me a list of watcher's profiles. Respond in this format:
[{
"gender": (F/M)
"age":(the age of the watcher)
"job":(the job of the watcher)
}]
Respond:
Now please generate:

Table 24: The profile prompt template in SoC.

Your task is to give me a list of {num added} person's profiles for twitter users. Respond in this format: [{{ "user name": "(str;The name of this user)", "user description":"(str;short and concise, a general description of this user, ordinary users or super large users and the topics this person interested in)" }}]
Now please generate: