CoS: Towards Optimal Event Scheduling via Chain-of-Scheduling

Yiming Zhao,[1,2] Jiwei Tang,[3] Shimin Di,[4,5] Libin Zheng,[1*] Jianxing Yu,[1] Jian Yin[1]

[1]School of Artificial Intelligence, Sun Yat-sen University
[2]Key Laboratory of Intelligent Assessment Technology for Sustainable Tourism, Ministry of Culture and Tourism, Sun Yat-sen University
[3]Shenzhen International Graduate School, Tsinghua University
[4]School of Computer Science and Engineering, Southeast University
[5]Key Laboratory of New Generation Artificial Intelligence Technology and Its Interdisciplinary Applications

Abstract

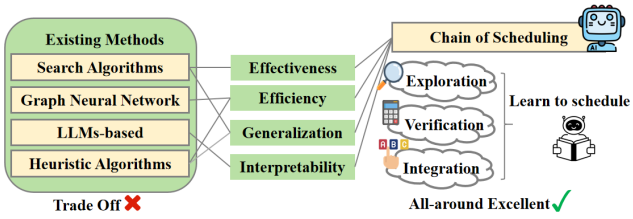
Recommending event schedules is a key issue in Event-based Social Networks (EBSNs) in order to maintain user activity. An effective recommendation is required to maximize the user's preference, subjecting to both time and geographical constraints. Existing methods face an inherent trade-off among efficiency, effectiveness, and generalization, due to the NP-hard nature of the problem. This paper proposes the **Chain-of-Scheduling (CoS)** framework, which activates the event scheduling capability of Large Language Models (LLMs) through a guided, efficient scheduling process. CoS enhances LLM by formulating the schedule task into three atomic stages, *i.e.*, *exploration*, *verification* and *integration*. Then we enable the LLMs to generate CoS autonomously via Knowledge Distillation (KD). Experimental results show that CoS achieves near-theoretical optimal effectiveness with high efficiency on three real-world datasets in a interpretable manner. Moreover, it demonstrates strong zero-shot learning ability on out-of-domain data.

Code — https://github.com/kiki123-hi/CoS

Introduction

As an emerging online social network, event-based social networks (EBSNs) enable users to organize and participate in offline social activities. Platforms such as Meetup, Eventbrite, and Douban City have more than 10 million monthly active users, playing a significant role in bridging the gap between online users to meet and communicate in the physical world. To maintain user activity, the core task behind these platforms is to effectively recommend events to users, which typically advises groups of events per user based on their interests. The recommended event group is usually organized as a schedule, allowing the user to participate offline in them from one to another (Figure 1(a) gives an example). Statistics show that Meetup has more than 16 million users who participate in over 300,000 events each month, thanks to its personalized event recommendation system (Cheng et al. 2021). Thus, effectively scheduling events for users is a key issue for EBSNs.

(a) A specific example of Event Scheduling.



(b) Challenges of Event Scheduling.

Figure 1: A illustration of event scheduling. (a) is a specific example of event scheduling. (b) denotes the challenges of event scheduling, *i.e.*, existing methods face a trade-off among effectiveness, efficiency, and interpretability while CoS can achieve all-round excellent via *exploration*, *verification*, and *integration*.

Current methods for event scheduling problem can be categorized into three main types based on their primary concerns: effectiveness, efficiency, and generalization.

Effectiveness-targeted methods aim to achieve higher schedule quality by precisely matching users' preferences subjecting to the complex constraints. Among these, search algorithms like grid search and dynamic programming (Zhang et al. 2023) find optimal solutions but face

*Corresponding author: zhenglb6@mail.sysu.edu.cn

at least quadratic time complexities, leading to its inefficiency for large problem inputs. In recent years, Large Language Models (LLMs) have emerged as an area of research in the planning domain, with prompt engineering techniques like Chain-of-Thought (CoT) achieving significant progress in various Natural Language Processing (NLP) tasks (Yang et al. 2024; DeepSeek-AI et al. 2025). Nevertheless, when it comes to event scheduling, even with long CoTs often suffer from redundancy, lack of focus, and overthinking (Chen et al. 2024; Sui et al. 2025; Su et al. 2025), leading to time-consuming exploration and invalid schedules (Figure 2(b) gives an example). Automated agent workflow generation methods (Zhang et al. 2025; Huang, Lipovetzky, and Cohn 2025; Ouyang et al. 2025; Xu et al. 2025) struggle to fully understand the complex constraints, making their planning behaviors abnormal or fall into ineffective loops (Valmeekam, Stechly, and Kambhampati 2024; Kambhampati 2024; Aghzal et al. 2025).

Efficiency-targeted methods focus on efficiently generating schedules, typically employing heuristic algorithms to hold a low computational complexity, such as greedy algorithms (Cheng et al. 2021) and genetic algorithms (Alhijawi and Awajan 2024). However, these methods often sacrifice effectiveness, and face lower schedule quality regarding users' preferences. Graph Neural Networks (GNNs) (Liu and Huang 2023; Chen, Thiébaux, and Trevizan 2024) are constrained by their small parameter scale and black-box nature, making it difficult to model complex event relationships. In addition, their lack of interpretability is also a serious concern for usage.

In terms of **generalization**, aforementioned methods are generally stable and yield consistent behaviors on unseen data, but limited to either effectiveness or efficiency. In contrast, GNNs struggle to learn the underlying logic and general rules behind the data, preventing them from effectively transferring existing knowledge and experience when faced with unseen data distributions, leading to poor generalization (Ma, Deng, and Mei 2021; Wu et al. 2024).

Event scheduling for EBSN, which requires considering both temporal and spatial constraints, is an NP-hard problem. As illustrated in Figure 1(b), the aforementioned methods inevitably face a hard trade off on certain aspects. This naturally leads to a research question: *How can we achieve effective and efficient event scheduling in an interpretable manner while maintaining strong zero-shot learning capabilities?*

To this end, we propose Chain-of-Scheduling (CoS), which specifically standardizes the scheduling task into three atomic stages, *exploration*, *verification*, and *integration*, while fully leveraging the interpretability of LLMs. Different from CoT, CoS is not a general-purpose thinking process. Instead, it provides a preset, rigorously structured guidance framework. It significantly boosts computational efficiency by formulating efficient reasoning paths for LLMs and minimizing unproductive inference steps, thus avoiding the degradation of CoT performance caused by redundant thinking and lack of clear direction. Specifically, the *exploration* step guides LLMs in targeted exploration, drastically narrowing the ineffective search space and enabling LLMs to more *efficiently* find high-quality candidate solutions. The *verification* and *integration* steps then guide LLMs to evaluate candidate solutions and make optimal choices, ensuring the *effectiveness* of the generated schedule while forming a complete, coherent, and *interpretable* scheduling solution. To enable LLMs to learn to schedule, we perform Knowledge Distillation (KD) (Gou et al. 2021). Specifically, we use search algorithms as teacher models, constructing the high quality solutions into a natural language format. Then we distill this scheduling knowledge into LLMs via Supervised Fine-tuning (SFT). Through this process, LLMs internalize rich spatiotemporal knowledge and complex constraint handling logic, allowing them to *generalize* to unseen datasets.

Our contributions are three-fold:

- We propose CoS, a framework that integrates *exploration*, *verification* and *integration* to achieve all-round excellence in event scheduling across effectiveness, efficiency, generalization, and interpretability.
- Through Knowledge Distillation (KD), We enable the LLMs to generate CoS autonomously, which guides LLMs to internalize spatiotemporal knowledge and complex constraint handling logic for event scheduling.
- Experimental results show our method significantly outperforms other existing methods in terms of efficiency and effectiveness, and demonstrates strong generalization ability on out-of-domain unseen datasets.

Preliminaries

Event Scheduling Data. An event refers to an offline activity that a user may participate in, with the i-th event denoted as e_i. All the announced events on the EBSN comprise a set as $E = \{e_i\}$. Each event $e_i = \langle loc_i, t_i^{start}, t_i^{end} \rangle$ is associated with a geographical location loc_i (where the event is held), and the starting & ending time t_i^{start}, t_i^{end}. Users give varying preferences over the events. For each user u_j, there is a utility score $s_{i,j}$ representing their preference for e_i. In practice, $s_{i,j}$ is typically computed according to the profiles of events and users, and each EBSN develops its own model for $s_{i,j}$ regarding the platform characteristics (Li et al. 2014; She, Tong, and Chen 2015; She et al. 2016). Thus, following existing works (Li et al. 2014; She, Tong, and Chen 2015; She et al. 2016; Cheng et al. 2017, 2021) , we focus on how to effectively recommending and scheduling the events, and treat $s_{i,j}$ as our input. Note that our developed model is orthogonal to the computation of $s_{i,j}$'s, and is applicable for all EBSNs.

Event Scheduling. Given an event set E and a user u_j, the recommendation task on EBSN is to generate an event sequence $T^* = \langle e_{i_1} \to e_{i_2} \to \cdots \rangle$ that maximizes the total utility score while retaining validity.

We first define the *feasible solution space* as:

$$\mathcal{T} = \left\{ T = \langle e_{i_1} \to e_{i_2} \to \dots \rangle \right\}. \quad (1)$$

$$\text{s.t. } \forall\, (e_i \to e_{i'}) \in T\,, \quad t_{i'}^{start} - t_i^{end} \geq t(loc_i, loc_{i'})\,,$$

where the function $t(\cdot)$ denotes the traveling time between the two locations.

Then, the optimal event sequence T^* is given by

$$T^* = \arg\max_{T \in \mathcal{T}} \sum_{e_i \in T} s_{i,j} \,. \tag{2}$$

Related Work

Large Language Models. The emergence of large language models has transformed the research paradigm in the Natural Language Processing (NLP) community (Tang et al. 2025a,b; Guo et al. 2025), especially the oversized large models such as DeepSeek and Qwen (DeepSeek-AI et al. 2025, 2024; Yang et al. 2024). Recently, the concept of test-time computation has gained popularity, with many studies employing chain of thought and reinforcement learning on top of base models to further scale up performance at test time, achieving impressive results across various tasks. However, when directly applied to planning tasks, these Oversized large models often perform poorly, even failing to complete simple actions (Valmeekam, Stechly, and Kambhampati 2024; Kambhampati 2024; Aghzal et al. 2025). In our event scheduling tests, Oversized large models often take more than ten minutes to process a planning scenario with fewer than 20 total events, only to produce incorrect results. *Therefore, the planning potential of LLMs remains largely untapped.*

Combinatorial Optimization. Although event scheduling is a NP-hard problem, there are still some Combinatorial Optimization that can find exact solutions, such as grid search and dynamic programming (Bastos et al. 2019). However, these methods have high time complexity, above quadratic level, and are unable to handle a large number of event inputs. There are also some heuristic algorithms, such as the greedy algorithm (Cheng et al. 2021) that searches for local optima and can solve event scheduling relatively quickly, and the genetic algorithm (Alhijawi and Awajan 2024) that searches for global optima but at a slower pace. Yet, the results of these methods often have a significant gap from the optimal solution. *Therefore, Combinatorial Optimization either face a trade-off between time efficiency and effectiveness or fail to achieve satisfactory performance in both aspects, making it difficult to realize optimal event scheduling.*

Traditional Deep Learning-based Methods. Event scheduling can be formalized as a weighted activity selection problem and modeled by GNNs. Prior works include heuristic graph search algorithms that learn domain-independent heuristic functions via GNNs (Chen, Thiébaux, and Trevizan 2024), and reinforcement learning-based GNNs that can model event scheduling as Markov decision processes to learn state representations and train agents for near-optimal decisions (Liu and Huang 2023). *However, these methods are limited by the relatively small parameter scale and modeling capacity of GNNs, leading to poor performance in event scheduling problems. Moreover, they lack interpretability, essentially operating as black boxes to select the optimal set of events that satisfy the constraints.*

Method

As shown in Figure 2 (b), raw LLMs are still weak in terms of generating high-quality event schedules. To upgrade LLMs for such a task, as shown in Figure 2 (a), we devise the Chain-of-Scheduling (CoS) framework, which treats it as a student model and teaches it to schedule with a chain of reasoning. In particular, CoS consists of two components. The first is CoS Construction component, preparing a set of chain-of-scheduling data (the reasoning and evidence for generating high-quality schedules), which would be used to teach/train the LLM in the second component. The second component is the CoS Knowledge Distillation component, which distills the constructed schedule knowledge chain into the LLM.

The Construction of CoS

We first describe how we construct the CoS data to be used in the Knowledge Distillation stage.

Chain-of-Scheduling (CoS) formalizes the schedule reasoning process as a composition of three atomic stages: 1) *Exploration*: exploring high-quality schedules, 2) *Verification*: verifying the utility score of each solution, and 3) *Integration*: integrate the current evidence and select the best solution.

Exploration. To mimic the human's reasoning for generating a satisfying schedule, this step first enumerates k high-quality schedules, which are valid in the first place. They could be either the exact top-k schedules or the approximated top-k ones because of the time complexity variance. In terms of the former, they are denoted as

$$T_{\text{top-}k} = \arg \text{top-}k_{T \in \mathcal{T}} \left(\sum_{e_i \in T} s_{i,j} \right) , \tag{3}$$

where $T_{\text{top-}k}$ is top-k candidate schedules.

In the offline training step, we employ grid search or dynamic programming (DP) (Zhang et al. 2023) to collect such top-k schedules for each training input instance. In this way, the effectiveness-targeted algorithms like DP serve as the teacher model while the LLM serves as the student model. The insight lies in: although search algorithms like dynamic programming may be computationally expensive, they reliably yield optimal or near-optimal solutions. This motivates us to take it as a knowledge source for teaching LLMs.

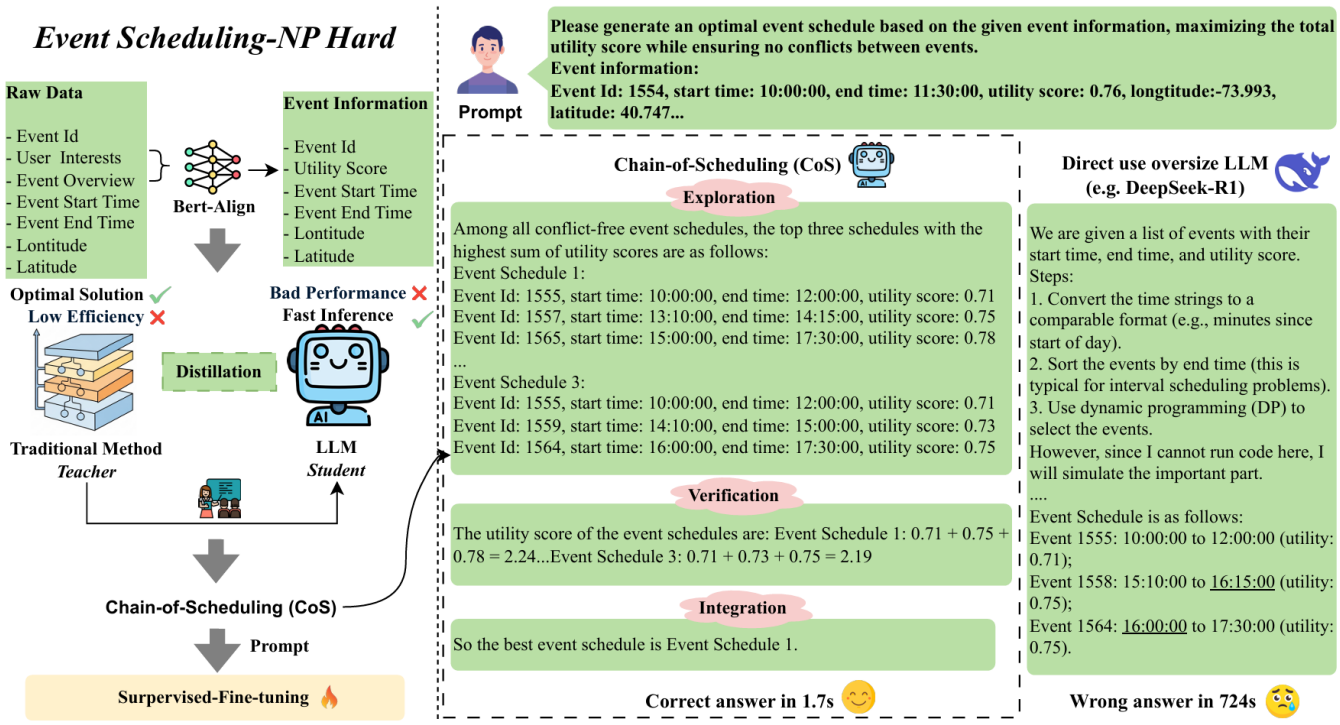
Verification. After exploration, we mimic a human's verification thinking by evaluating each solution's quality, *i.e.*, explicitly computing the utility score for each schedule. For example, if a schedule contains two events with utility scores 2 and 3, the verification step generates the reasoning trace: "2 + 3 = 5". This can be formalized as

$$v(T) = \sum_{e_i \in T} s_{i,j} \,, \tag{4}$$

where $v(S)$ is the verified total utility score of schedule $s_{i,j}$.

Integration. Once exploration and verification are complete, we have multiple near-optimal schedules along with their verified utility scores. The final step is to pick the maximum:

$$T^* = \text{argmax}_{T \in T_{\text{top-}k}} v(T) \,, \tag{5}$$



(a) Teach the LLM to generate CoS.

(b) A specific running example.

Figure 2: **The illustration of the Chain-of-Scheduling (CoS).** (a) denotes the basic process of teaching the LLM to generate CoS. Distillation CoS from Combinatorial Optimization, then supervised fine-tune (SFT) the (prompt, CoS) pairs, enabling the LLM to autonomously generate CoS. (b) is a specific running example. Small LLM (*e.g.*, Qwen2.5-7B-Instruct) can quickly arrive at the correct answer via CoS, while directly using a oversized LLM (*e.g.*, DeepSeek-R1) will fall into an overly lengthy and time-consuming thinking process, ultimately leading to an incorrect answer.

where T^* is the optimal event schedule with maximum utility score.

CoS Knowledge Distillation

The core goal of this phase is to distill the structured reasoning capability of the CoS framework into LLMs. This is achieved by training LLMs to autonomously generate CoS traces given input event sets and user contexts, framed as knowledge distillation: meticulously constructed CoS traces (from algorithms like dynamic programming) serve as the teacher model, while LLMs act as student models mimicking the teacher's step-by-step reasoning. Through SFT, this high-quality reasoning knowledge is transferred to LLMs. Leveraging LLMs' inherent efficient reasoning (e.g., parallel decoding) and inference acceleration frameworks (e.g., vLLM), SFT-tuned LLMs can reproduce CoS reasoning with ultra-low latency, enabling them to inherit traditional methods' high-quality planning capabilities while addressing computational inefficiency.

CoS Alignment. We construct a specialized SFT dataset $D_{\text{SFT}} = \{(x_i, y_i^{\text{CoS}})\}_{i=1}^{N}$ to bridge CoS reasoning and LLM text generation. Each input x_i is a complete event scheduling problem instance, containing an event set, user context, and preference information. The target output y_i^{CoS} is the complete CoS reasoning trace generated for x_i by search algorithm-based CoS construction (as shown in Figure 2 (b)). Natural language rationales in y_i^{CoS} bridge the gap between symbolic-logical constructs and LLM-compatible textual reasoning traces.

SFT Objective Function. The goal of SFT is to maximize the likelihood of LLMs generating correct CoS traces, achieved by minimizing token-level cross-entropy loss over the entire CoS. For a single instance (x_i, y_i^{CoS}), the loss is:

$$\mathcal{L}_{\text{SFT}}^{(i)}(\theta) = -\sum_{t=1}^{L_i} \log P_{\text{LLM}}(y_{i,t}^{\text{CoS}} \mid x_i, y_{i,<t}^{\text{CoS}}; \theta)\,, \quad (6)$$

where L_i is the token length of y_i^{CoS}, and $y_{i,t}^{\text{CoS}}, y_{i,<t}^{\text{CoS}}$ denote the t-th token and preceding tokens of y_i^{CoS}, respectively. The overall SFT objective across the dataset is:

$$\mathcal{L}_{\text{SFT}}(\theta) = \mathbf{E}_{(x,y^{\text{CoS}})\sim\mathcal{D}_{\text{SFT}}}\left[\mathcal{L}_{\text{SFT}}^{(i)}(\theta)\right]\,. \quad (7)$$

Minimizing $\mathcal{L}_{\text{SFT}}(\theta)$ trains LLMs to accurately predict CoS sequence tokens, replicating structured reasoning and distilling traditional methods' planning capabilities.

Schedule Post-processing

As above, we distill the spatiotemporal knowledge essential for event scheduling to LLM, to achieve both effectiveness

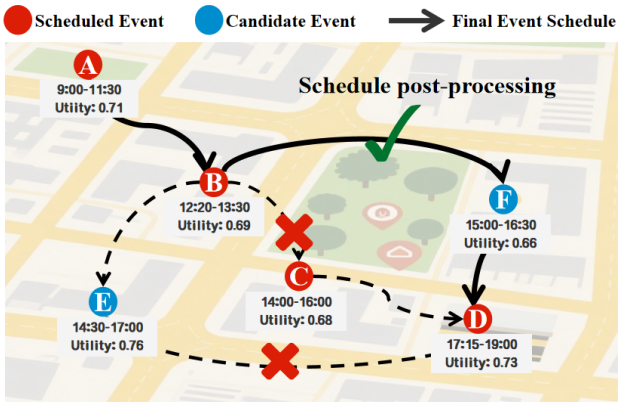


Figure 3: A post-processing example.

and efficiency. However, since the hallucination issue could not yet be eliminated for the state-of-the-art LLMs (Ji et al. 2023; Chakraborty, Ornik, and Driggs-Campbell 2025), the fine-tuned LLM with CoS is also likely to produce schedules that are not all valid. As later shown by the experiment results, though LLM equipped CoS already achieves a large improvement in terms of the event conflicts (two events are not compatible in terms of time or distance), a valid schedule is not always guaranteed. We thereby propose a light-weight post-processing step, which further confirms the validity of the schedule.

As shown in Figure 3, the original generated schedule is $A \rightarrow B \rightarrow C \rightarrow D$, while event B and C are too close in time while far away in terms of distance, preventing the user from reaching C in time after finishing B. To eliminate this conflict, our post-processing would perform a local search which anchors on the first conflicted event to find substitution for the successor event. In the example, it finds the event F which is compatible with both B and D, leading to the new schedule $A \rightarrow B \rightarrow F \rightarrow D$ with a slight utility loss.

Experiments

In this section, we attempt to answer the following four research questions (RQs):

- **RQ1:** How does Chain-of-Scheduling (CoS) perform compared to baselines?
- **RQ2:** How effective is the zero-shot learning capability of CoS?
- **RQ3:** How effective are the various components within CoS?
- **RQ4:** How does the parameter k in the *Exploration* step affect CoS?

Experimental Setting

Dataset Description. To evaluate the effectiveness of the proposed model across different urban computing scenarios, we crawl data from the Meetup website from 2-year period on three cities, *i.e.*, New York, Washington, and London. We eventually obtain 74411, 81395, 218773 events and 45854, 44742, and 22381 users for the cities respectively. In this way, we evaluate the methods across different urban scenarios, investigating their robustness. For each city, the data is further divided into training and test sets in a 4:1 ratio. To form a training/testing sample, we randomly draw a user and a date, and then obtain the corresponding events on the date. An input instance is then formulated as recommending an event schedule for the user on the date (from 9 am to 9 pm). In particular, for testing, we repeat more than 1600 evaluations and report the average performance.

An event includes three features: event ID, the time window, and the event description. A user contains features: user ID, interests, and the user's attended events. For each evaluation input, the locations and time features for the events could be directly obtained from the dataset. Since the platforms do not directly disclose users' preferences over events, we simulate the utility score between an event and a user. We use BERT (Devlin et al. 2019) to align the event/user descriptions and then compute their semantic similarity, obtaining a utility score ranging from 0 to 1. Note that our method is orthogonal to any utility formulation, and our simulation in this experiment effectively captures users' ground-truth participation over events.

Base Models. To fully the validate effectiveness and superiority of our method, we conduct experiments with two types of base models: one is the dense model Qwen2.5-7B-Instruct (Yang et al. 2024), and the other is the sparse model Mistral-7B-Instruct-v0.3 (Nadhavajhala and Tong 2024) based on mixed-of-experts (MoE).

Hyperparameters Settings. In the construction phase of Chain-of-Scheduling (CoS), we select the top k event schedules with the highest utility scores, where k is set to 3 by default.

Our method employs lightweight Low-Rank Adaptation, *i.e.*, LoRA training approach (Hu et al. 2022), with specific parameter settings as follows: the maximum length of the model is set to 32,768, the learning rate is set to 1e-5, the number of epochs is set to 3, the batch size is set to 2, the alpha value of lora is set to 16, and the rank value is set to 8. The entire training process is carried out on two NVIDIA A800-SXM4-80GB GPUs.

Baselines. We compare with three categories of methods: Pre-trained LLMs, combinatorial optimization, and traditional deep learning-based methods. We provide a description for each baseline below.

Standard pretrained LLMs:

- Qwen2.5-Plus (Yang et al. 2024): an optimized version of Qwen2.5 by Alibaba, with strong language capabilities, outperforming DeepSeek-V2.5.
- Qwen2.5-Max (Yang et al. 2024): an MoE model by Alibaba with over 100 billion parameters and 100k context length.
- DeepSeek-V3 (DeepSeek-AI et al. 2024): an open-source model by DeepSeek, with 671 billion parameters and 128k token context window.
- DeepSeek-R1 (DeepSeek-AI et al. 2025): a reasoning model by DeepSeek, trained via reinforcement learning.

Methods	NewYork			Washington			London		
	Utility Score	Latency (seconds)	Conflict Rate (%)	Utility Score	Latency (seconds)	Conflict Rate (%)	Utility Score	Latency (seconds)	Conflict Rate (%)
Standard pretrained LLMs									
DeepSeek-R1	2.97	785	66	2.99	496	98	3.01	750	99
DeepSeek-V3	2.45	48.7	90	2.71	46.5	89	2.68	35.4	94
Qwen-Max	2.44	141	97	2.63	197	98	2.81	145	97
Qwen-Plus	2.23	34.3	98	2.43	38.1	99	2.11	42.7	99
Combinatorial Optimization									
Grid Search	3.73	$> 10^3$	–	4.02	$> 10^3$	–	5.17	$> 10^3$	–
Dynamic Programming	3.73	14.2	–	4.02	8.58	–	5.17	24.6	–
Greedy	1.95	**0.01**	–	2.31	**0.01**	–	2.51	**0.01**	–
Genetic Algorithm	1.14	3.79	–	1.07	4.14	–	1.12	9.86	–
Deep learning-based Methods									
GOOSE	3.15	59.2	–	3.02	58.8	–	2.93	60.5	–
GNN-DRL	2.85	0.44	–	2.64	0.45	–	3.18	4.23	–
LLMs with multi-step reasoning									
DeepSeek-V3 + Chain-of-Thought	2.48	172	86	2.58	104	84	2.54	108	88
Qwen-Plus + Chain-of-Thought	2.16	30.1	94	2.44	38.1	98	2.19	50.6	96
Qwen2.5-7B + Chain-of-Thought	2.14	2.32	99	2.32	4.72	99	2.41	7.54	99
Mistral-7B + Chain-of-Thought	2.12	2.27	98	2.19	2.57	97	2.22	5.07	96
Qwen2.5-7B + Chain-of-**Scheduling**	3.37	1.29	15	3.57	1.39	22	4.50	3.36	41
Mistral-7B + Chain-of-**Scheduling**	**3.44**	1.84	**9.8**	**3.64**	1.94	**15**	**4.65**	4.79	**19**

Table 1: Performance comparison of different methods. We **bold** the best, underline the second best and double underline the third among the efficiency-aware methods (i.e., except the time-costly exact solutions)

Method \ Utility Score	Washington	London
Deep learning-based Methods		
GOOSE	3.32	2.85
GNN-DRL	2.60	3.04
LLMs with multi-step reasoning		
Qwen2.5-7B + Chain-of-**Scheduling**	3.65	4.22
Mistral-7B + Chain-of-**Scheduling**	3.47	3.87

Table 2: Generalization test results

For LLM-based methods, since their generated schedule may be invalid, we adopt the same post-processing proposed in Section 4.3 to ensure fairness.

Combinatorial Optimization:

- Greedy (Li et al. 2014; She et al. 2016; Cheng et al. 2017, 2021): it selects events with the highest utility score without conflicts.
- Grid Search: it enumerates all solutions to find the optimal conflict-free task set.
- Dynamic Programming (Zhang et al. 2023): it breaks down complex problems to maximize total utility score.
- Genetic Algorithm (Alhijawi and Awajan 2024): it optimizes event sequences using natural selection.

Traditional deep learning-based methods:

- GOOSE (Chen, Thiébaux, and Trevizan 2024): it uses GNNs to guide classical planning search.
- GNN-DRL (Liu and Huang 2023): it solves DJSSP using GNNs and DRL, minimizing total processing time.

Evaluation Metrics. We comprehensively evaluate the performance of various methods in handling event scheduling from three perspectives. **Utility Score** represents the user's average preference for each given sequence of events, with higher values indicating better. **Latency** evaluates the running time of each method. **Conflict Rate** measures the proportion of events conflicting with each other in a schedule. Remember that to ensure a valid schedule, for two adjacent events in the sequence, their time windows should not overlap and retain a gap allowing the user to physically travel from one to another (otherwise resulting in a conflict). In our evaluation, we find that LLM-based methods cannot ensure 0% conflicts, so we have a post-processing step as mentioned before. Nevertheless, we still report their conflict rates before the post-processing step, serving as a complementary metric to explain the methods' performances.

Main Results (RQ1)

In this section, we report the comparison results across three different cities: New York, Washington, and London. We can draw four key findings:

- **Standard pretrained LLMs generally suffer low utility scores and high latency.** The reason for the low utility scores could be reflected by their conflict rates, which are dramatically larger than other methods. Even with chain-of-thoughts, they are still struggling for generating schedules where events are compatible with each other.

Methods	Utility Score	Latency (seconds)	Conflict Rate (%)
Default	3.37	1.29	–
w/o *Exploration*	3.15	0.51	58
w/o *Verification*	3.31	1.04	19
w/o *Integration*	3.26	1.13	24
w/o *Post-processing*	3.32	1.31	15

Table 3: Ablation study comparing our methods with baselines on the New York dataset. The conflict rate refers to the model's scheduling results prior to the removal of conflicts

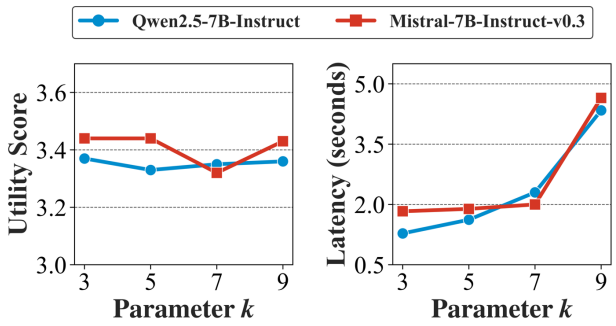
They typically have higher latency due to API calls.

- **Combinatorial Optimization fail to achieve both satisfying latency and utility.** Although grid search and dynamic programming can obtain optimal solutions, their time complexity is above the quadratic level, leading to long delays. The greedy algorithm is quick but has a utility score far from the optimum. Genetic algorithm, on the other hand, is weak in either aspects.
- **Graph Neural Networks-based methods are not satisfactory either.** GNN-DRL is fast but achieve inferior utilities, achieving $61 \sim 76\%$ of the optimum. GOOSE is better on New York and Washington, but suffers high latency. In particular, the ratio between their utility and the optimum decay over the three datasets, from New York to London. This indicates their performances become worse on larger inputs.
- **CoS maintains low planning latency while significantly surpassing other approximation methods in utility across all datasets.** The utility of CoS on all datasets remain above 90% of the optimum, far higher than other approximation methods. The planning latency is kept within 2 seconds on datasets with fewer events, i.e., New York and Washington, and within 5 seconds on London. The ourstanding peformance of CoS is attributed to its low conflict rate compared to others.

Zero-Shot Event Scheduling Performance (RQ2)

In this section, we examine the predictive performance of the proposed model in zero-shot scenarios, with the results presented in Table 2. Specifically, we train the methods on the New York dataset and test them on the Washington and London datasets.

The results in Table 2 highlight CoS's exceptional performance in event scheduling on unseen cities London and Washington, with its Utility Score being up to 50% higher than other methods. CoS's zero-shot learning capability is benefited from its semantic understanding for scheduling events, via our knowledge distillation of the high-quality schedules. Although we only train on the New York dataset, the model indeed learns the universal and transferable spatiotemporal scheduling semantic knowledge, allowing it to interpret and manage the data from unseen cities. It is worth noting that the New York dataset actually has significantly fewer events than Washington and London's, which further demonstrates the effective learning paradigm of CoS.



(a) Utility Score. (b) Latency.

Figure 4: Hyperparameter Sensitivity Analysis.

Ablation Study (RQ3)

In this section, we examine the role of each step in CoS in unleashing the potential of LLMs. Specifically, as shown in Table 3, we respectively ablate the three steps of CoS, *i.e.*, *Exploration*, *Verification*, and *Integration*, to observe their impacts. W/o *Exploration* means expecting the model to directly generate the optimal schedule without comparing to other candidates. W/o *Verification* means removing the verification process. W/o *Integration* means randomly selecting a schedule from the candidate schedules, and w/o *Post-processing* means directly removing conflicting events.

We find that removing any component significantly degrades the performance of CoS, which fully demonstrates the effectiveness of each part. Among them, *Exploration* is the most important (with a 7% drop), indicating that the initial construction of the solution space is crucial for ultimately deriving the optimal event schedule.

Hyperparameter Sensitivity Analysis (RQ4)

In the CoS framework, there is a tunable parameter k, the number of candidate solutions in the *exploration* phase. As shown in Figure 4, we conduct a hyperparameter sensitivity analysis on two different backbones, *i.e.*, Qwen2.5-7B and Mistral-7B. The results indicate that the average utility scores remain stable at a high level across different k values, while the inference time increases over a larger k, which is expected given the enlarged search space. Overall, we choose $k = 3$, which already offers enough demonstration knowledge to the model on what is a high-quality schedule, while retaining a low latency.

Conclusion

This paper introduces CoS, which aims to stimulate the planning capabilities of LLMs. It fully stimulates the semantic understanding ability of large models for event scheduling. We organize CoS into text that LLMs can understand through text alignment, and distill the high-quality scheduling capabilities of search algorithms into LLMs via SFT, thereby efffciently and effectively completing event scheduling tasks.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Nos. 62472455, 62506075, U22B2060, 62276279); the Key-Area Research and Development Program of Guangdong Province (No. 2024B0101050005); the Research Foundation of the Science and Technology Plan Project of Guangzhou City (Nos. 2023B01J0001, 2024B01W0004); Guangdong Basic and Applied Basic Research Foundation (No. 2024B1515020032).

References

Aghzal, M.; Plaku, E.; Stein, G. J.; and Yao, Z. 2025. A Survey on Large Language Models for Automated Planning. *CoRR*, abs/2502.12435.

Alhijawi, B.; and Awajan, A. 2024. Genetic algorithms: theory, genetic operators, solutions, and applications. *Evol. Intell.*, 17(3): 1245–1256.

Bastos, L. S. L.; Marchesi, J. F.; Hamacher, S.; and Fleck, J. L. 2019. A mixed integer programming approach to the patient admission scheduling problem. *Eur. J. Oper. Res.*, 273(3): 831–840.

Chakraborty, N.; Ornik, M.; and Driggs-Campbell, K. R. 2025. Hallucination Detection in Foundation Models for Decision-Making: A Flexible Definition and Review of the State of the Art. *ACM Comput. Surv.*, 57(7): 188:1–188:35.

Chen, D. Z.; Thiébaux, S.; and Trevizan, F. W. 2024. Learning Domain-Independent Heuristics for Grounded and Lifted Planning. In *AAAI*, 20078–20086. AAAI Press.

Chen, X.; Xu, J.; Liang, T.; He, Z.; Pang, J.; Yu, D.; Song, L.; Liu, Q.; Zhou, M.; Zhang, Z.; Wang, R.; Tu, Z.; Mi, H.; and Yu, D. 2024. Do NOT Think That Much for 2+3=? On the Overthinking of o1-Like LLMs. *CoRR*, abs/2412.21187.

Cheng, Y.; Yuan, Y.; Chen, L.; Giraud-Carrier, C. G.; and Wang, G. 2017. Complex Event-Participant Planning and Its Incremental Variant. In *ICDE*, 859–870. IEEE Computer Society.

Cheng, Y.; Yuan, Y.; Chen, L.; Giraud-Carrier, C. G.; Wang, G.; and Li, B. 2021. Event-Participant and Incremental Planning over Event-Based Social Networks. *IEEE Trans. Knowl. Data Eng.*, 33(2): 474–488.

DeepSeek-AI; Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; et al. 2025. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *CoRR*, abs/2501.12948.

DeepSeek-AI; Liu, A.; Feng, B.; Xue, B.; Wang, B.; Wu, B.; et al. 2024. DeepSeek-V3 Technical Report. *CoRR*, abs/2412.19437.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*, 4171–4186. Association for Computational Linguistics.

Gou, J.; Yu, B.; Maybank, S. J.; and Tao, D. 2021. Knowledge Distillation: A Survey. *Int. J. Comput. Vis.*, 129(6): 1789–1819.

Guo, H.; Zhu, J.; Di, S.; Shi, W.; Chen, Z.; and Xu, J. 2025. DioR: Adaptive Cognitive Detection and Contextual Retrieval Optimization for Dynamic Retrieval-Augmented Generation. In *ACL (1)*, 2953–2975. Association for Computational Linguistics.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*. OpenReview.net.

Huang, S.; Lipovetzky, N.; and Cohn, T. 2025. Planning in the Dark: LLM-Symbolic Planning Pipeline Without Experts. In *AAAI*, 26542–26550. AAAI Press.

Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y.; Madotto, A.; and Fung, P. 2023. Survey of Hallucination in Natural Language Generation. *ACM Comput. Surv.*, 55(12): 248:1–248:38.

Kambhampati, S. 2024. Can Large Language Models Reason and Plan? *CoRR*, abs/2403.04121.

Li, K.; Lu, W.; Bhagat, S.; Lakshmanan, L. V. S.; and Yu, C. 2014. On social event organization. In *KDD*, 1206–1215. ACM.

Liu, C.; and Huang, T. 2023. Dynamic Job-Shop Scheduling Problems Using Graph Neural Network and Deep Reinforcement Learning. *IEEE Trans. Syst. Man Cybern. Syst.*, 53(11): 6836–6848.

Ma, J.; Deng, J.; and Mei, Q. 2021. Subgroup Generalization and Fairness of Graph Neural Networks. In *NeurIPS*, 1048–1061.

Nadhavajhala, S.; and Tong, Y. 2024. Rubra-Mistral-7B-Instruct-v0.3.

Ouyang, C.; Yue, L.; Di, S.; Zheng, L.; Yue, L.; Pan, S.; Yin, J.; and Zhang, M. 2025. Code2MCP: Transforming Code Repositories into MCP Services. *CoRR*, abs/2509.05941.

She, J.; Tong, Y.; and Chen, L. 2015. Utility-Aware Social Event-Participant Planning. In *SIGMOD Conference*, 1629–1643. ACM.

She, J.; Tong, Y.; Chen, L.; and Cao, C. C. 2016. Conflict-Aware Event-Participant Arrangement and Its Variant for Online Setting. *IEEE Trans. Knowl. Data Eng.*, 28(9): 2281–2295.

Su, J.; Healey, J.; Nakov, P.; and Cardie, C. 2025. Between Underthinking and Overthinking: An Empirical Study of Reasoning Length and correctness in LLMs. *CoRR*, abs/2505.00127.

Sui, Y.; Chuang, Y.; Wang, G.; Zhang, J.; Zhang, T.; Yuan, J.; Liu, H.; Wen, A.; Zhong, S.; Chen, H.; and Hu, X. B. 2025. Stop Overthinking: A Survey on Efficient Reasoning for Large Language Models. *CoRR*, abs/2503.16419.

Tang, J.; Xu, J.; Lu, T.; Zhang, Z.; Zhao, Y.; Hai, L.; and Zheng, H. 2025a. Perception Compressor: A Training-Free Prompt Compression Framework in Long Context Scenarios. In *NAACL (Findings)*, 4093–4108. Association for Computational Linguistics.

Tang, J.; Zhang, Z.; Wu, S.; Ye, J.; Bai, L.; Wang, Z.; Lu, T.; Chen, J.; Hai, L.; Zheng, H.; and Kim, H. 2025b. GMSA: Enhancing Context Compression via Group Merging and Layer Semantic Alignment. *CoRR*, abs/2505.12215.

Valmeekam, K.; Stechly, K.; and Kambhampati, S. 2024. LLMs Still Can't Plan; Can LRMs? A Preliminary Evaluation of OpenAI's o1 on PlanBench. *CoRR*, abs/2409.13373.

Wu, Q.; Nie, F.; Yang, C.; Bao, T.; and Yan, J. 2024. Graph Out-of-Distribution Generalization via Causal Intervention. In *WWW*, 850–860. ACM.

Xu, S.; Zhang, J.; Di, S.; Luo, Y.; Yao, L.; Liu, H.; Zhu, J.; Liu, F.; and Zhang, M. 2025. RobustFlow: Towards Robust Agentic Workflow Generation. *CoRR*, abs/2509.21834.

Yang, A.; Yang, B.; Zhang, B.; Hui, B.; Zheng, B.; et al. 2024. Qwen2.5 Technical Report. *CoRR*, abs/2412.15115.

Zhang, J.; Xiang, J.; Yu, Z.; Teng, F.; Chen, X.; Chen, J.; Zhuge, M.; Cheng, X.; Hong, S.; Wang, J.; Zheng, B.; Liu, B.; Luo, Y.; and Wu, C. 2025. AFlow: Automating Agentic Workflow Generation. In *ICLR*. OpenReview.net.

Zhang, Y.; Zou, L.; Liu, Y.; Ding, D.; and Hu, J. 2023. A brief survey on nonlinear control using adaptive dynamic programming under engineering-oriented complexities. *Int. J. Syst. Sci.*, 54(8): 1855–1872.