

Bridging Auxiliary Constraints to Resolve Instruction Following in Large Reasoning Models

Zhengyi Zhao^{1,7}, Shubo Zhang^{1,7}, Huimin Wang³, Zezhong Wang^{1,7}, Yutian Zhao³,
Yefeng Zheng⁴, Binyang Li^{2*}, Yulan He^{5,6}, Kam-Fai Wong^{1,7}, Xian Wu^{3*}

¹ The Chinese University of Hong Kong ² University of International Relations

³ Tencent Jarvis Lab ⁴ Westlake University ⁵ King's College London ⁶ The Alan Turing Institute

⁷ Ministry of Education Key Laboratory of High Confidence Software Technologies, CUHK

{zyzhao, kfwong}@se.cuhk.edu.hk, byli@uir.edu.cn, kevinxwu@tencent.com

Abstract

Large Reasoning Models (LRMs) have demonstrated impressive capabilities in many tasks, yet they struggle with reliably following multiple instructions, either by failing to satisfy individual constraints or by struggling to balance competing constraints simultaneously. We formalize this challenge as the Constraint Adherence Problem (CAP). This paper introduces a novel framework that addresses CAP by representing instructions as a structured knowledge graph of constraints. Our approach, Constraint Relationship Graph Completion (CRGC), explicitly models relationships between constraints, identifies adherence challenges, and discovers “bridge constraints” that help the model better focus on and reconcile requirements. Bridge constraints act as auxiliary instructions that make primary constraints more salient and compatible. Unlike existing approaches that enhance instruction following through general training methods, CRGC specifically improves constraint satisfaction by leveraging the model’s own knowledge to create better pathways for generation. Experiments across three popular instruction following datasets demonstrate that our approach reduces constraint violations by 39% compared to standard prompting while maintaining reasoning abilities of large reasoning models.

1 Introduction

Large Reasoning Models (LRMs) have shown remarkable abilities in responding to instructions, yet they frequently struggle with fully satisfying all requirements in a given prompt (Honovich et al., 2023; Peng et al., 2024; Wei et al., 2025; Luo et al., 2025). This challenge shows in two key ways: (1) models sometimes overlook certain

constraints entirely (Zhang et al., 2024; Shi et al., 2024b; Pei et al., 2025), and (2) models often fail to balance multiple constraints that appear to conflict (Pan et al., 2023; Jiang et al., 2024; Li et al., 2025). For example, when asked to “write a detailed technical explanation about quantum computing within 200 words,” an LRM might produce text that is either too detailed (violating word limit) or too simplified (violating technical detail). In other cases, a model might completely overlook a constraint such as “enclose your answer with double quotation marks” even when there’s no apparent conflict with other requirements.

We formalize this widespread challenge as the *Constraint Adherence Problem (CAP)*, which represents a significant barrier to reliable deployment of LRMs in real-world applications (Ouyang et al., 2022; Zhao et al., 2025b). While researchers have developed various approaches to improve instruction following through supervised fine-tuning (Wu et al., 2023; Du and Chilton, 2023), reinforcement learning from human feedback (Shi et al., 2024a; Liu et al., 2025), and prompt engineering techniques (Yin et al., 2023; Srikanth et al., 2024; Held et al., 2025), these methods don’t specifically address how models recognize, prioritize, and reconcile multiple constraints. Instead, they implicitly assume that better general capabilities will naturally lead to better constraint satisfaction, an assumption that may not always hold in practice.

Our key insight is that effective constraint satisfaction requires explicitly modeling how constraints relate to each other and to the overall task. To this end, we introduce a structured approach where each instruction is broken down into specific constraints that form a relationship graph. In this graph, nodes represent individual constraints, and edges represent how constraints interact with each other, supporting, interfering with, or acting independently of one another.

This graph-based representation reveals that

* Corresponding Author

LRMs struggle most when constraints are either insufficiently salient or perceived as contradictory, termed as constraint interference. Based on this insight, we develop the *Constraint Relationship Graph Completion* (CRGC) framework with three main components: (1) **Constraint Graph Construction** that maps relationships between decomposed instruction constraints; (2) **Adherence Challenge Detection** that identifies overlooked or conflicting constraints; and (3) **Bridge Constraint Discovery** that introduces auxiliary instructions to connect or reconcile problematic constraints. We define *bridge constraints* as intermediate instructions that address conceptual gaps causing model failures by creating logical connections between conflicting or disconnected primary constraints. Importantly, we express bridge constraints in natural language rather than categorizing them into predefined types, preventing artificial restrictions on model behavior and avoiding the deployment of multiple bridges for a single constraint pair. For example, when a model struggles to write text that is both “comprehensive” and “within 200 words,” a bridge constraint like “use structured sections with bullet points for key details” provides a concrete strategy to satisfy both requirements. Our approach leverages the model’s knowledge to enhance constraint satisfaction without requiring additional training or complex prompting strategies.

We evaluated our approach across instruction following benchmarks. Experimental results demonstrate that CRGC reduces constraint violations by 39% compared to standard prompting and by 27.7% compared to Chain-of-Thought (CoT) prompting, while maintaining or improving output quality as assessed by human evaluators. Notably, unlike existing instruction-following methods that typically sacrifice performance on general reasoning tasks after specialization, our approach shows no degradation on standard reasoning benchmarks. This advantage stems from CRGC’s non-invasive nature, it enhances constraint satisfaction without modifying model parameters and adaptively determines when bridge constraints are beneficial, automatically omitting them when unnecessary. The primary contributions of this paper are:

- A formal characterization of the Constraint Adherence Problem in LRMs, primarily focusing on constraint oversight and constraint interference problems

- A graph-based framework for representing constraint relationships and identifying adherence challenges, along with an effective method for discovering bridge constraints.
- Empirical evidence demonstrating great improvements in constraint satisfaction across diverse datasets without requiring model re-training, while maintaining the reasoning abilities.

2 Related Works

Recent research has focused on evaluating and enhancing instruction following capabilities in LLMs. Evaluation benchmarks such as [Jiang et al. \(2024\)](#) and [Qin et al. \(2024\)](#) assess hierarchical and complex instruction following, while [Wu et al. \(2025\)](#) proposed metrics for cross-context consistency. Parameter-update approaches have improved instruction adherence through techniques like reinforcement learning from human feedback ([Ouyang et al., 2022](#)) and instruction tuning ([Wei et al., 2022a](#); [Chung et al., 2024](#); [Longpre et al., 2023](#)). However, these methods typically treat instruction following as a monolithic skill without explicitly modeling relationships between potentially conflicting constraints.

Multi-attribute steering represents another approach to control specific aspects of LLM outputs. [Dathathri et al. \(2020\)](#) and [Krause et al. \(2021\)](#) introduced plug-and-play frameworks for controlling attributes like toxicity and sentiment, while [Subramani et al. \(2022\)](#) identified latent directions in embedding space corresponding to specific attributes. More recent work by [Wang et al. \(2024\)](#) developed techniques for simultaneously controlling multiple attributes through vector composition. While these approaches share our goal of satisfying multiple requirements, our work differs in several key aspects: (1) we address arbitrary constraints with complex interdependencies rather than predefined independent attributes, (2) we leverage the model’s reasoning capabilities through bridge constraints instead of modifying internal representations, and (3) we explicitly model constraint relationships to identify which ones need reconciliation, making our approach more adaptive to diverse instruction types where constraints may be implicit, conflicting, or hierarchical.

A parallel line of research focuses on post-hoc iterative refinement to improve constraint ad-

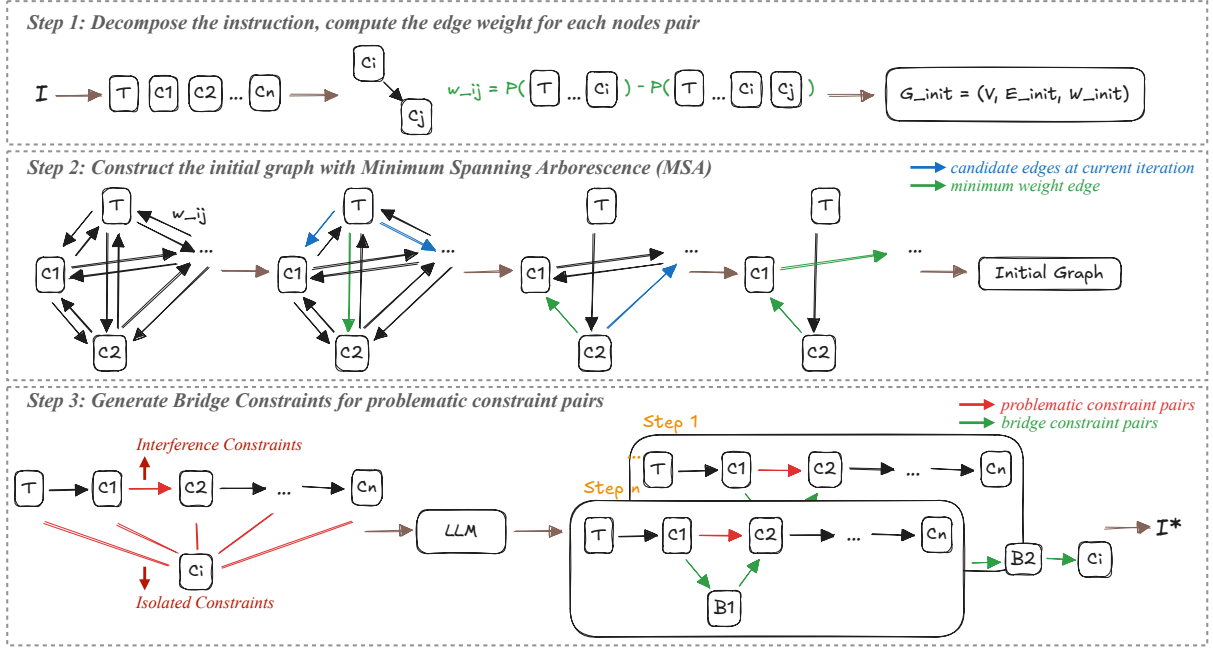


Figure 1: Overview of the Constraint Relationship Graph Completion (CRGC) framework. Step 1: The input instruction I is decomposed into individual constraints $\{C_1, C_2, \dots, C_n\}$, and edge weights w_{ij} are computed based on the leave-one-out probabilities to form the initial graph G_{init} . Step 2: The Minimum Spanning Arborescence (MSA) algorithm (Edmonds et al., 1967) constructs an optimized constraint graph by iteratively selecting minimum weight edges (green) from candidate edges (blue) to connect all constraint nodes to the task node T . Step 3: Problematic constraint pairs (with red arrows) are identified, including interference constraints and isolated constraints. Bridge constraints B_1, B_2 are then generated using an LLM to address these problematic pairs, resulting in an enhanced instruction I^* that improves constraint adherence.

herence. Frameworks such as DeCRIM (Ferah et al., 2024) and Divide-Verify-Refine (Zhang et al., 2025) employ a multi-stage pipeline where an initial draft is generated, evaluated against constraints, and subsequently refined to correct identified violations. While these iterative correction methods improve upon zero-shot prompting, they operate reactively. Because they lack an explicit, holistic model of how constraints interact, fixing one violation often inadvertently triggers a new violation in a previously satisfied constraint. In contrast, our CRGC framework operates proactively. By constructing a comprehensive relationship graph prior to generation, CRGC identifies inherent conflicts and deploys bridge constraints to structurally reconcile them during the initial forward pass. This graph-based foresight not only yields higher ultimate satisfaction rates for conflicting constraints but fundamentally bypasses the computationally expensive reactive correction.

3 Methodology

In this section, we present our Constraint Relationship Graph Completion (CRGC) framework in

detail. We first formalize the Constraint Adherence Problem (CAP) and introduce the concept of directed constraint relationships. Then, we describe our three-component approach: (1) Constraint Graph Construction, (2) Adherence Challenge Detection, and (3) Bridge Constraint Discovery.

3.1 Problem Formulation

Given a language model M that maps instructions I to output text $y \in \mathcal{Y}$, we define a constraint C as a specific requirement that the output text must satisfy. An instruction I typically contains a core task R and multiple constraints $C = \{C_1, C_2, \dots, C_n\}$. The Constraint Adherence Problem (CAP) occurs when a model fails to satisfy all constraints in an instruction, either by overlooking certain constraints entirely or by failing to balance multiple constraints that appear to conflict.

3.1.1 Constraint Satisfaction

For a given constraint C_i and output text y , we define a satisfaction function $S(C_i, y) \in [0, 1]$ by

prompting an evaluation Large Language Model (LLM) as the judge, where 1 indicates perfect satisfaction and 0 indicates complete violation. For binary constraints (e.g., “enclose answer in quotes”), $S(C_i, y)$ is either 0 or 1. For continuous constraints (e.g., “make explanation detailed”), $S(C_i, y)$ can take intermediate values. The overall constraint satisfaction for an instruction is given by the bottleneck performance:

$$S_{\text{total}}(I, y) = \min_{C_i \in \mathcal{C}} S(C_i, y). \quad (1)$$

We use the minimum rather than the average because satisfying most constraints while completely violating one remains a failure in strict instruction following.

3.1.2 Directed Constraint Influences

Constraint relationships in autoregressive generation are inherently asymmetric. We characterize three fundamental directed relationships based on how satisfying constraint C_i influences the expected satisfaction of constraint C_j . Let $\mathbb{E}[S(C_j, y)]$ denote the baseline expected satisfaction, and $\mathbb{E}[S(C_j, y) \mid S(C_i, y) \geq \tau]$ ¹ denote the conditional expectation given that C_i is highly satisfied, where τ denotes the relaxation rate to evaluate whether or not the instruction can be regarded as followed.

Definition 3.1 (Enhancing Influence). *Constraint C_i enhances C_j if satisfying C_i strictly increases the the expected satisfaction of C_j :*

$$\mathbb{E}[S(C_j, y) \mid S(C_i, y) \geq \tau] > \mathbb{E}[S(C_j, y)]. \quad (2)$$

Definition 3.2 (Interfering Influence). *Constraint C_i interferes with C_j if satisfying C_i decreases the expected satisfaction of C_j , representing a perceived conflict:*

$$\mathbb{E}[S(C_j, y) \mid S(C_i, y) \geq \tau] < \mathbb{E}[S(C_j, y)]. \quad (3)$$

Definition 3.3 (Neutral / Independent Influence). *Constraint C_i has a neutral influence on C_j if satisfying C_i does not meaningfully affect the satisfaction of C_j :*

$$|\mathbb{E}[S(C_j, y) \mid S(C_i, y) \geq \tau] - \mathbb{E}[S(C_j, y)]| \leq \varepsilon, \quad (4)$$

where ε is a small margin of independence. Note that this is a directed property of the edge $C_i \rightarrow C_j$, not a symmetric global independence.

¹Here, we set τ to strictly satisfy the i -th constraint.

3.2 Constraint Relationship Graph Completion Framework

Our CRGC framework systematically addresses the CAP through three sequential steps: constructing a constraint relationship graph, detecting adherence challenges, and discovering appropriate bridge constraints. This framework leverages the model’s own capabilities to enhance constraint satisfaction without requiring parameter updates.

3.2.1 Constraint Graph Construction

The first step decomposes instructions into individual constraints to construct a weighted directed graph representing their influences. Given an instruction I , we construct an initial fully connected directed graph $G_{\text{init}} = (V, E_{\text{init}}, W_{\text{init}})$, where the vertex set $V = \{C_1, C_2, \dots, C_n\} \cup \{\mathbf{T}\}$ comprises all constraints plus the overall task objective. The edge set $E_{\text{init}} = \{(C_i, C_j) \mid C_i, C_j \in V, i \neq j\}$ contains directed connections.

To maximize overall constraint satisfaction, we want to find a generation pathway that maximizes enhancing relationships and avoids interfering ones. Thus, we define the edge weight $w_{ij} \in W_{\text{init}}$ as the *cost* (or risk of violation) of attempting to satisfy C_j after C_i . This is quantified by the relative drop in expected satisfaction:

$$w_{ij} = \frac{\mathbb{E}[S(C_j, y)] - \mathbb{E}[S(C_j, y) \mid S(C_i, y) \geq \tau]}{\mathbb{E}[S(C_j, y)]}. \quad (5)$$

To ensure statistically meaningful estimates, we generate N candidate outputs using a non-zero temperature to capture diverse generation paths².

Under this formulation, if C_i enhances C_j , the cost w_{ij} is negative (beneficial). If C_i interferes with C_j , the cost w_{ij} is positive (detrimental). We then apply the Minimum Spanning Arborescence (MSA) algorithm (Edmonds et al., 1967) to find a directed spanning tree rooted at $\mathbf{R}$ with minimum total cost. This directly aligns the MSA objective with our goal: producing a directed tree that minimizes constraint violation risk. Formally, we seek the tree $T^* = (V, E_T, W_T)$ such that:

$$T^* = \arg \min_{T' \in \mathcal{T}} \sum_{(i,j) \in E_{T'}} w_{ij}, \quad (6)$$

where $\mathcal{T}$ is the set of all possible directed spanning trees rooted at $\mathbf{R}$.

²Here we set N equals to 11.

3.2.2 Adherence Challenge Detection

The second step analyzes the optimized tree T^* to identify constraint relationships that still present challenges. We categorize edges in the tree based on their cost weights:

$$\text{Category}(i, j) = \begin{cases} \text{Enhance,} & \text{if } w_{ij} < -\delta, \\ \text{Independent,} & \text{if } |w_{ij}| \leq \delta, \\ \text{Interfere,} & \text{if } w_{ij} > \delta, \end{cases} \quad (7)$$

where δ is an empirically determined positive threshold³. Interfering edges represent constraints that the model perceives as contradictory despite the optimized path, while independent edges represent disjoint constraints that the model fails to cognitively connect, leading to overlooked requirements.

We identify the set of problematic edges $E_{\text{prob}} \subset E_T$ as those categorized as interfering or independent. These edges become candidates for bridge constraint insertion.

3.2.3 Bridge Constraint Discovery

The final step addresses the identified adherence challenges by generating intermediate natural language instructions. For each problematic edge $(C_i, C_j) \in E_{\text{prob}}$, we generate a bridge constraint B_{ij} using the language model M :

$$B_{ij} = \begin{cases} \mathcal{R}(C_i, C_j), & \text{if } w_{ij} > \delta \text{ (interfering)} \\ \mathcal{C}(C_i, C_j), & \text{if } |w_{ij}| \leq \delta \text{ (independent)}, \end{cases} \quad (8)$$

where $\mathcal{R}(\cdot, \cdot)$ and $\mathcal{C}(\cdot, \cdot)$ are reconciliation and connection prompt templates, respectively.

For interfering constraints, $\mathcal{R}(\cdot, \cdot)$ guides the model to resolve perceived conflicts. For independent constraints, $\mathcal{C}(\cdot, \cdot)$ establishes explicit relationships. For example, given interfering constraints “be concise” (C_i) and “provide detailed explanations” (C_j), the model might generate the bridge “focus on key information with precise language while ensuring all necessary details are included.”

We incorporate these bridges to create an enhanced structure $G_{\text{enhanced}} = (V_{\text{enhanced}}, E_{\text{enhanced}}, W_{\text{enhanced}})$, where $V_{\text{enhanced}} = V \cup \{B_{ij} \mid (C_i, C_j) \in E_{\text{prob}}\}$. Problematic edges are replaced with bridge-mediated connections: $E_{\text{enhanced}} = (E_T \setminus E_{\text{prob}}) \cup \{(C_i, B_{ij}), (B_{ij}, C_j) \mid (C_i, C_j) \in E_{\text{prob}}\}$.

³Here we set δ equals to 0.3 as default.

Finally, we augment the original instruction by seamlessly context-integrating the bridge constraints:

$$I_{\text{enhanced}} = I \oplus \{B_{ij} \mid (C_i, C_j) \in E_{\text{prob}}\}, \quad (9)$$

where $\oplus$ denotes the concatenation and formatting of the new constraints into the prompt payload. This targeted intervention resolves adherence failures dynamically, allocating computational overhead strictly to the nodes that exhibit conceptual friction.

4 Experiments

4.1 Experimental Settings

Datasets. We evaluate CRGC on three instruction following benchmarks and three general reasoning benchmarks to demonstrate both improved constraint adherence and preserved reasoning capabilities.

Our evaluation employs three comprehensive instruction following benchmarks. IFEval (Zhou et al., 2023) provides a comprehensive benchmark with 541 instructions across 25 categories, designed to evaluate various aspects of instruction following abilities. ComplexBench (Wen et al., 2024) contains 1,150 complex, multi-constraint instructions that require models to balance competing requirements. FollowBench (Jiang et al., 2024) features 820 instruction-response pairs with explicit constraints and evaluation criteria focusing on instruction handling capabilities.

To ensure our method preserves reasoning abilities, we utilize three standard reasoning benchmarks. MMLU (Hendrycks et al., 2021) serves as a multitask benchmark covering 57 subjects across STEM, humanities, social sciences, and more. GSM8K (Cobbe et al., 2021) contains 8,500 high-quality grade school math problems requiring multi-step reasoning. BIG-Bench Hard (BBH) (Suzgun et al., 2023), consists of 23 challenging tasks selected from BIG-Bench focusing on complex reasoning.

Evaluation Metrics. For comprehensive evaluation, we employ several complementary metrics across different dimensions. To ensure rigorous assessment and prevent evaluation circularity, our evaluation pipeline processes constraints according to their type. First, for strictly quantitative and structural constraints (e.g., word count limits, specific formatting, presence of keywords),

we utilize deterministic Python scripts (e.g., exact token counters and regex) to strictly bypass the known limitations of LLMs on exact counting tasks. Second, for semantic constraints requiring qualitative judgment, we employ a decoupled evaluation framework. The evaluation model is strictly isolated from the generation model (e.g., using Claude-3.7-Sonnet to evaluate GPT-4o outputs, and vice versa) to eliminate self-preference bias.

We evaluate constraint adherence using two primary metrics. Constraint Satisfaction Rate (CSR) measures the proportion of constraints fully satisfied in a response:

$$\text{CSR} = \frac{1}{|I|} \sum_{i=1}^{|I|} \frac{1}{|C_i|} \sum_{j=1}^{|C_i|} \mathbb{1}[S(C_{ij}, y_i) \geq \tau], \quad (10)$$

where I is the set of all instructions, C_i is the set of constraints for instruction i , $S(C_{ij}, y_i)$ is the satisfaction score assigned by our evaluation pipeline for constraint j of instruction i , and τ is the satisfaction threshold. Weighted Constraint Satisfaction (WCS) provides a more nuanced evaluation by accounting for partial satisfaction of continuous constraints:

$$\text{WCS} = \frac{1}{|I|} \sum_{i=1}^{|I|} \frac{1}{|C_i|} \sum_{j=1}^{|C_i|} S(C_{ij}, y_i), \quad (11)$$

where $S(C_{ij}, y_i) \in [0, 1]$ represents the degree of satisfaction.

To assess overall task effectiveness, we use multiple performance indicators. Task Completion Rate (TCR) measures the proportion of tasks completed successfully:

$$\text{TCR} = \frac{1}{|I|} \sum_{i=1}^{|I|} S_{\text{total}}(I_i, y_i), \quad (12)$$

where $S_{\text{total}}(I_i, y_i)$ is the overall satisfaction score for instruction i . Standard accuracy metrics are applied for reasoning tasks, while F1 Score is used for tasks with precision-recall tradeoffs.

Baselines. We compare CRGC against state-of-the-art models and instruction following methods.

Our evaluation includes both open-source and closed-source models. For open-source models, we utilize Qwen2.5-72B (Alibaba, 2025), Llama-3.1-70B (Meta, 2024), and Mixtral-8x22B (Mistral, 2024). The closed-source models in our com-

parison include GPT-4o (OpenAI, 2024), Claude-3.7-Sonnet-Thinking (Anthropic, 2025), Gemini-2.5-Pro (Google, 2025).

We compare against several established instruction following techniques. Standard Prompting (SP) serves as our baseline, using direct instruction input without additional techniques (Wei et al., 2022b). CoT prompts models to show step-by-step reasoning (Wei et al., 2022b). Self-Consistency (SC) generates multiple responses and selects the most consistent output (Wang et al., 2023). Self-Reflection (SR) prompts models to reflect on their outputs against requirements (Ji et al., 2023). Constraint Optimization Prompting (COP) implements explicit optimization for constraint satisfaction (Do et al., 2024). Think-to-Think (T²) solve the problem by reflecting multiple samples to model for reference (Zhao et al., 2025a).

Implementation Details. We implemented automated constraint extraction using GPT-4o decomposition. We computed the Minimum Spanning Arborescence using NetworkX (Hagberg et al., 2008) to identify critical constraint relationships. Claude-3-Opus generated candidate bridge constraints with temperature equals 0.7, which were filtered through the evaluation LLM to avoid introducing new conflicts. Bridge constraints were integrated using a template that maintained instruction coherence while making constraint relationships explicit. We used vLLM (Kwon et al., 2023) for open-source model inference and APIs for closed-source models. Each experiment was repeated ten times with different random seeds, reporting average performance with standard deviations⁴. Appendix A presents the detailed implementation and comprehensive reproducibility.

4.2 Main Results

Table 1 demonstrates CRGC’s superior performance against all baseline methods across instruction following benchmarks. For closed-source models, CRGC yields significant CSR improvements over Standard Prompting, CoT, and SR with similar patterns on ComplexBench and FollowBench. Open-source models show consistent gains, with CRGC improving Qwen2.5-72B’s CSR by 11.9% over Standard Prompting. Notably, CRGC outperforms specialized instruction following methods (SC, T², and COP) even on

⁴The code can be found in [Here](#).

Table 1: Performance comparison on three instruction following benchmarks. Best performance for each model is highlighted in **bold**. CSR, WCS, TCR stand for constraint-level, weighted-constraint-level, and task-level performance, respectively. The baselines SP, CoT, SR denotes Standard Prompting, Chain-of-Thoughts, and Self-Reflection, respectively.

Method	IFEval			ComplexBench			FollowBench		
	CSR	WCS	TCR	CSR	WCS	TCR	CSR	WCS	TCR
<i>Closed-source Models</i>									
GPT-4o + SP	78.9±1.2	82.5±1.0	80.3±1.1	66.7±1.5	70.8±1.3	68.1±1.4	73.2±1.3	77.3±1.1	74.5±1.2
GPT-4o + CoT	82.7±0.9	86.1±0.8	84.2±0.9	71.5±1.2	75.4±1.0	72.8±1.1	77.1±1.0	80.9±0.9	78.3±1.0
GPT-4o + SR	84.5±0.8	87.8±0.7	86.1±0.8	73.2±1.0	77.1±0.9	74.6±1.0	78.9±0.9	82.7±0.8	80.1±0.9
GPT-4o + CRGC	88.7±0.6	91.3±0.5	89.6±0.7	78.6±0.8	82.5±0.7	79.8±0.9	83.8±0.7	87.4±0.6	84.9±0.8
Claude-3.7-Sonnet + SP	79.5±1.2	83.1±0.9	81.2±1.1	67.4±1.4	71.6±1.2	68.8±1.3	74.3±1.3	78.2±1.1	75.7±1.2
Claude-3.7-Sonnet + CoT	83.6±0.9	87.0±0.7	85.2±0.8	72.3±1.1	76.2±0.9	73.7±1.0	78.4±1.0	82.1±0.8	79.5±0.9
Claude-3.7-Sonnet + SR	85.7±0.7	88.9±0.6	87.3±0.7	74.1±0.9	78.0±0.8	75.5±0.9	80.2±0.8	84.0±0.7	81.4±0.8
Claude-3.7-Sonnet + CRGC	90.2±0.5	92.7±0.4	91.0±0.6	79.8±0.7	83.6±0.6	81.0±0.8	85.1±0.6	88.6±0.5	86.3±0.7
Gemini-2.5-Pro + SP	78.2±1.3	81.9±1.0	79.8±1.2	65.8±1.5	70.0±1.3	67.2±1.4	72.7±1.4	76.8±1.2	74.0±1.3
Gemini-2.5-Pro + CoT	82.0±1.0	85.5±0.8	83.6±0.9	70.6±1.2	74.5±1.0	72.0±1.1	76.5±1.1	80.3±0.9	77.8±1.0
Gemini-2.5-Pro + SR	83.8±0.8	87.1±0.7	85.5±0.8	72.3±1.0	76.3±0.9	73.8±1.0	78.2±0.9	82.0±0.8	79.5±0.9
Gemini-2.5-Pro + CRGC	88.0±0.6	90.7±0.5	89.1±0.7	77.5±0.8	81.4±0.7	78.9±0.9	83.0±0.7	86.7±0.6	84.2±0.8
<i>Open-source Models</i>									
Qwen2.5-72B + SP	74.3±1.5	78.0±1.3	75.8±1.4	62.5±1.7	66.9±1.5	63.9±1.6	69.2±1.6	73.5±1.4	70.6±1.5
Qwen2.5-72B + CoT	78.1±1.2	81.7±1.0	79.4±1.1	66.8±1.4	70.9±1.2	68.2±1.3	72.9±1.3	77.1±1.1	74.2±1.2
Qwen2.5-72B + SR	80.3±1.0	83.7±0.9	81.5±1.0	68.9±1.2	73.0±1.0	70.3±1.1	74.8±1.1	78.9±0.9	76.0±1.0
Qwen2.5-72B + CRGC	85.5±0.8	88.9±0.7	86.7±0.9	74.8±1.0	78.7±0.8	76.1±1.0	80.4±0.9	84.2±0.7	81.5±0.9
Llama-3.1-72B + SP	75.1±1.4	78.8±1.2	76.5±1.3	63.4±1.6	67.7±1.4	64.7±1.5	70.0±1.5	74.2±1.3	71.4±1.4
Llama-3.1-72B + CoT	79.0±1.1	82.5±0.9	80.3±1.0	67.8±1.3	71.9±1.1	69.1±1.2	73.8±1.2	77.9±1.0	75.0±1.1
Llama-3.1-72B + SR	81.2±0.9	84.5±0.8	82.4±0.9	69.7±1.1	73.8±0.9	71.0±1.0	75.6±1.0	79.6±0.8	76.8±0.9
Llama-3.1-72B + CRGC	86.3±0.7	89.5±0.6	87.4±0.8	75.6±0.9	79.4±0.7	76.8±0.9	81.2±0.8	84.9±0.6	82.3±0.8
Mixtral-8x22B + SP	72.4±1.6	76.1±1.4	73.9±1.5	60.3±1.8	64.5±1.6	61.7±1.7	67.2±1.7	71.4±1.5	68.5±1.6
Mixtral-8x22B + CoT	76.0±1.3	79.6±1.1	77.3±1.2	64.5±1.5	68.6±1.3	65.8±1.4	70.8±1.4	74.9±1.2	72.1±1.3
Mixtral-8x22B + SR	78.1±1.1	81.5±0.9	79.3±1.0	66.4±1.3	70.5±1.1	67.7±1.2	72.5±1.2	76.6±1.0	73.8±1.1
Mixtral-8x22B + CRGC	83.4±0.9	86.7±0.8	84.5±1.0	72.2±1.1	76.1±0.9	73.5±1.1	78.3±1.0	82.0±0.8	79.4±0.9

Table 2: Balance between instruction following and reasoning capabilities. The **gray cells** indicate performance degradation compared to the base model. IF and Reasoning denotes Instruct Following datasets and Reasoning datasets. H-Mean denotes Harmonic mean.

Method	IF (TCR Avg.)	Reasoning (Avg.)	Balance (H-Mean)	Method	IF (TCR Avg.)	Reasoning (Avg.)	Balance (H-Mean)
<i>GPT-4o</i>				<i>Llama-3.1-72B</i>			
Base (SP)	74.3	88.7	80.9	Base (SP)	70.9	84.2	77.0
+ CoT	78.4 (+4.1)	90.0 (+1.3)	83.8	+ CoT	74.8 (+3.9)	86.3 (+2.1)	80.1
+ SR	80.3 (+6.0)	89.5 (+0.8)	84.7	+ SR	76.7 (+5.8)	85.4 (+1.2)	80.8
+ CRGC	84.8 (+10.5)	90.2 (+1.5)	87.4	+ CRGC	82.2 (+11.3)	86.4 (+2.2)	84.2
+ Fine-tuned	84.4 (+10.1)	82.4 (-6.3)	83.4	+ Fine-tuned	83.1 (+12.2)	77.5 (-6.7)	80.2
<i>Claude-3.7-Sonnet</i>				<i>Mixtral-8x22B</i>			
Base (SP)	75.2	88.1	81.1	Base (SP)	68.0	81.4	74.1
+ CoT	79.5 (+4.3)	89.6 (+1.5)	84.2	+ CoT	71.7 (+3.7)	83.5 (+2.1)	77.1
+ SR	81.4 (+6.2)	89.0 (+0.9)	85.0	+ SR	73.6 (+5.6)	82.7 (+1.3)	77.9
+ CRGC	86.1 (+10.9)	89.7 (+1.6)	87.9	+ CRGC	79.1 (+11.1)	83.6 (+2.2)	81.3
+ Fine-tuned	87.1 (+11.9)	81.5 (-6.6)	84.2	+ Fine-tuned	79.9 (+11.9)	74.3 (-7.1)	77.0

GPT-4o. These improvements in both CSR and WCS metrics validate our approach for mitigating constraint oversight and interference problems through explicit constraint relationship modeling.

Table 2 reveals CRGC’s ability to reconcile instruction following with reasoning capabilities. While traditional instruction fine-tuning methods degrade reasoning abilities by 6-7% despite en-

Table 3: Ablation studies on GPT-4o showing the contribution of each component in CRGC. CSR: Constraint Satisfaction Rate, TCR: Task Completion Rate, Avg. Overhead: Average computational overhead compared to standard prompting.

Method	IFEval		ComplexBench		FollowBench		Avg. Overhead
	CSR	TCR	CSR	TCR	CSR	TCR	
CRGC (Full)	88.7±0.6	89.6±0.7	78.6±0.8	79.8±0.9	83.8±0.7	84.9±0.8	24.5%
w/o Bridge Constraints	84.5±0.8	85.7±0.9	73.2±1.0	74.5±1.1	78.9±0.9	80.1±1.0	17.2%
w/o Relationship Weights	83.6±0.9	84.8±1.0	72.4±1.1	73.8±1.2	78.0±1.0	79.3±1.1	19.8%
w/o MSA Optimization	86.1±0.7	87.2±0.8	75.9±0.9	77.1±1.0	80.9±0.8	82.1±0.9	23.1%
w/o Adherence Challenge Detection	83.0±1.0	84.2±1.1	71.8±1.2	73.1±1.3	77.4±1.1	78.7±1.2	12.3%
Only Explicit Constraints	80.5±1.1	81.9±1.2	69.0±1.3	70.4±1.4	75.1±1.2	76.5±1.3	8.1%
Only First-order Relationships	84.2±0.9	85.4±1.0	72.9±1.1	74.3±1.2	78.6±1.0	79.9±1.1	18.7%
Manual Bridge Constraints	87.0±0.7	88.2±0.8	76.8±0.9	78.1±1.0	82.0±0.8	83.2±0.9	24.5%

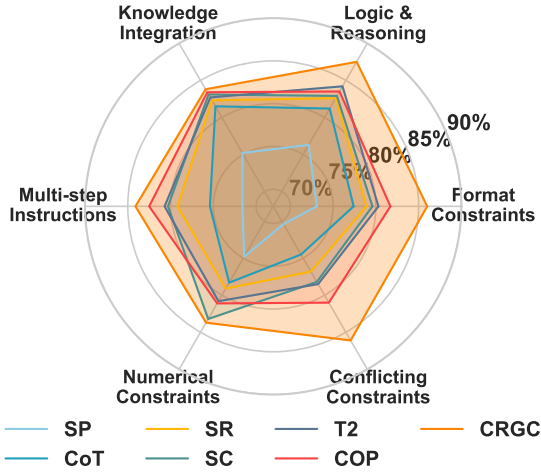


Figure 2: Analysis of CRGC performance across various constraint types showing CRGC’s consistent advantage, particularly for conflicting constraints and format requirements.

hancing instruction adherence, CRGC achieves superior instruction following while maintaining or slightly improving reasoning performance. This balanced performance stems from CRGC’s constraint-focused strategy that enhances instruction satisfaction without parameter modification, as confirmed by the harmonic mean metric.

4.3 Ablation Studies

Table 3 presents ablation studies using GPT-4o to evaluate each CRGC component. When bridge constraints are replaced with direct constraint connection, performance drops significantly, demonstrating their role in resolving conflicts. Without relationship weights, in other words, treating all relationships equally, or replacing MSA optimization with random spanning tree selection, the

model struggles to identify critical constraint relationships. Applying CRGC to all instructions instead of being based on adherence challenge detection results reduces efficiency without improving performance. The results also reveal that relying solely on explicit constraints or only first-order relationships in the experimental settings degrades outcomes, which confirms the value of comprehensive constraint modeling. Manual bridge constraints perform well but sometimes it would be very hard to find the appropriate bridge constraints for LLM resulting in the performance drop. But our CRGC use LLM to find bridge constraints for itself which is a more reasonable way. These results validate our design choices in balancing performance and computational efficiency.

4.4 Analysis of Constraint Relationships

Our CRGC is robust for various constraint type. Our analysis in Figure 2 shows CRGC consistently outperforms all baseline methods across diverse constraint categories. CRGC excels particularly in handling conflicting constraints and format requirements where other methods struggle. While specialized baselines may perform well in specific areas, none matches CRGC’s versatility across the constraint spectrum. More cases and details can be found in Appendix A.1.

Our CRGC is more efficient across multi-turn scenarios. While sophisticated reasoning frameworks improve performance, their computational efficiency is a critical consideration for real-world deployment. To evaluate this, we analyze a multi-turn iterative refinement protocol. Specifically, this is not an independent sampling (Best-of- n) approach; rather, it is a stateful, iterative interaction. If a model fails to satisfy all constraints

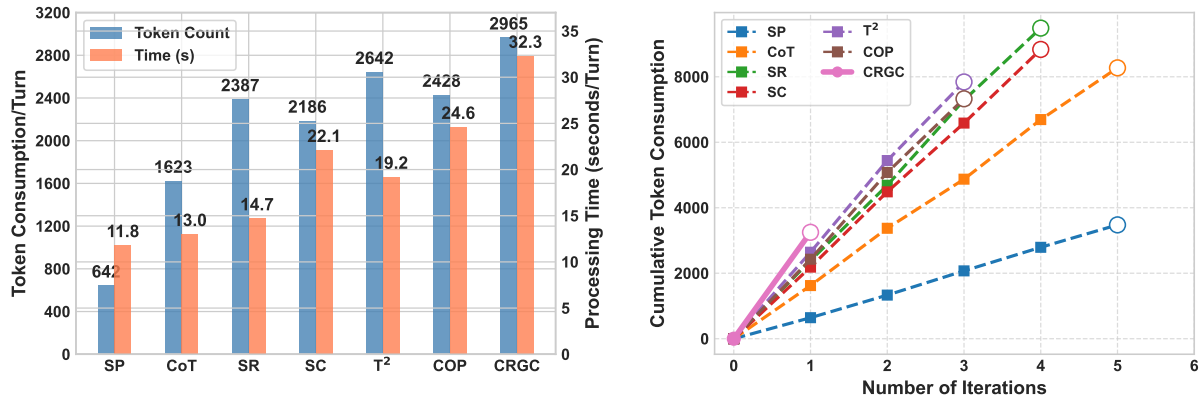


Figure 3: Computational efficiency comparison showing token consumption and processing time for different reasoning methods. The left shows results for a single turn, and the right shows cumulative computational costs across multiple turns required to satisfy all constraints.

on turn t , the context is extended with explicit feedback from the evaluation pipeline detailing the specific violated constraints, and the model is prompted for a corrected attempt at turn $t + 1$. We cap this process at a maximum of $n = 6$ generations per instruction.

Figure 3 (left) presents the token consumption and processing time across different methods on GPT-4o for a single initial turn. Our analysis reveals that CRGC has a higher single-turn latency compared to baseline methods due to the overhead of graph construction and edge weight computation. However, this single-turn comparison is incomplete. Figure 3 (right) illustrates the cumulative computational costs across the iterative multi-turn process required to reach full constraint satisfaction.

While baseline methods have lower per-turn token costs, they struggle to reconcile conflicts and typically require 4.8 iterations on average (out of the maximum $n = 6$) to reach full constraint satisfaction through trial and error. In contrast, CRGC resolves complex constraint relationships upfront, succeeding in just 1.2 turns on average. On instruction sets with high constraint interference, although CRGC requires 37% more tokens for the initial turn, its ability to drastically reduce the average number of required refinement turns yields a 67% reduction in total cumulative token consumption and a 58% reduction in overall processing time. This empirically validates our approach: a slightly higher upfront computational investment for substantially better first-pass instruction following eliminates the costly, repetitive back-and-forth refinement cycles that plague conventional

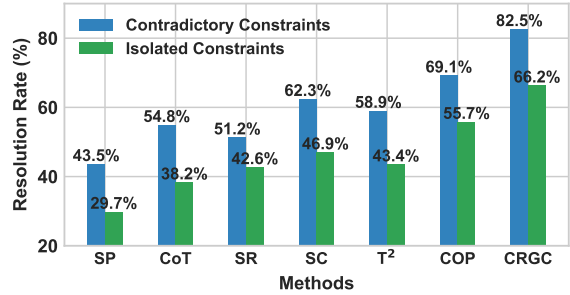


Figure 4: Comparative analysis of different methods on constraint adherence problem (CAP) resolution, highlighting CRGC’s superior ability to address both conflict constraints and isolated constraints that other methods struggle with.

prompting methods.

Analysis on CAP solving. We analyze how different reasoning methods handle problematic constraint resolution, examining conflict and isolated constraints which account for 27.3% and 31.8% of the benchmarks respectively. Figure 4 shows that Standard prompting struggles with both constraint types, while CoT and SR show moderate improvements with inconsistent patterns. SC and T² deliver further gains but exhibit volatility across constraint types. COP performs substantially better, yet our proposed CRGC method outperforms all alternatives for both conflict and isolated constraints. This performance gap demonstrates CRGC’s superior ability to systematically identify, represent, and resolve complex constraint relationships that conventional prompting methods fail to address.

Table 4: Comprehensive empirical comparison between proactive graph completion (CRGC) and reactive iterative refinement (DeCRIM) using GPT-4o. **Conf. Res.** denotes the conflict resolution rate. The **Avg-T** denotes the average inference turns required to reach terminal satisfaction, which is lower $\downarrow$ representing better. The Δ denotes the absolute gain. The results demonstrate that while DeCRIM improves upon Standard Prompting through costly multi-turn refinement, CRGC achieves strictly superior constraint satisfaction and conflict resolution in a fraction of the computational turns.

Methodology	IFEval Benchmark				ComplexBench				FollowBench			
	CSR	TCR	Conf. Res.	Avg-T $\downarrow$	CSR	TCR	Conf. Res.	Avg-T $\downarrow$	CSR	TCR	Conf. Res.	Avg-T $\downarrow$
SP	78.9	80.3	41.2%	1.00	66.7	68.1	28.5%	1.00	73.2	74.5	35.8%	1.00
DeCRIM	86.2	87.5	68.4%	3.85	75.1	76.8	54.2%	4.20	81.0	82.4	61.5%	4.05
CRGC (Ours)	88.7	89.6	84.6%	1.15	78.6	79.8	73.9%	1.28	83.8	84.9	80.2%	1.22
Δ	+2.5	+2.1	+16.2%	-2.70	+3.5	+3.0	+19.7%	-2.92	+2.8	+2.5	+18.7%	-2.83

Table 5: Comprehensive alignment analysis between automated evaluators and human annotators across 1,000 sampled constraints, separated by constraint type. The high correlation on deterministic checks (scripts) and semantic checks (decoupled LLMs) confirms the reliability of our evaluation framework and rules out self-preference bias.

Constraint Category	Coupled LLM (Self-Eval)			Single LLM Judge			Decoupled Framework (Ours)		
	Acc. (%)	Cohen’s κ	FPR (%)	Acc. (%)	Cohen’s κ	FPR (%)	Acc. (%)	Cohen’s κ	FPR (%)
Length (Word/Char Count)	68.4	0.42	21.3	71.2	0.48	18.6	100.0	1.00	0.0
Formatting (Markdown/JSON)	82.1	0.65	12.4	84.5	0.71	10.1	99.8	0.99	0.1
Keyword Inclusion	89.3	0.78	6.2	91.0	0.81	5.4	100.0	1.00	0.0
Semantic Requirements	85.6	0.72	8.9	86.4	0.74	8.2	92.1	0.85	3.4
Logical/Relational Rules	81.2	0.64	14.5	82.8	0.68	12.7	89.4	0.80	5.2
Overall Average	81.3	0.64	12.6	83.1	0.68	11.0	96.2	0.92	1.7

4.5 Comparison with Verify-and-Repair Strategies

To empirically validate the theoretical advantages of proactive graph completion over reactive refinement, we benchmarked CRGC directly against DeCRIM (Ferraz et al., 2024), a state-of-the-art verify-and-repair framework. DeCRIM utilizes a sophisticated pipeline that drafts, verifies individual constraints, and refines outputs over multiple turns (capped at 6 iterations in our replication).

Rather than relying on isolated qualitative case studies, Table 4 presents a comprehensive, large-scale quantitative analysis across all three benchmarks using the GPT-4o backbone. We assess not only the standard Constraint Satisfaction Rate (CSR) and Task Completion Rate (TCR), but also introduce two highly targeted metrics: the Conflict Resolution Rate (the percentage of diametrically opposed constraints successfully balanced) and the Average Inference Turns required to reach terminal satisfaction.

The empirical data in Table 4 shows the exact failure mode of verify-and-repair frameworks. While DeCRIM manages to brute-force general constraint satisfaction (CSR) to highly competitive levels, it falters significantly on the *Conflict*

Resolution Rate. Because DeCRIM treats constraint violations in isolation during its verification step, correcting one conflicting constraint frequently breaks its counterpart, leading to iteration exhaustion.

Conversely, CRGC excels precisely in these high-complexity scenarios. By mapping the interference edges in the constraint graph and synthesizing bridge constraints *before* generation begins, CRGC achieves up to a 19.7% absolute improvement in resolving diametrically opposed constraints compared to DeCRIM. Furthermore, it achieves this superior task completion while reducing the required inference turns by nearly 70%, proving that proactive structural foresight is fundamentally more capable and efficient than reactive iterative repair.

4.6 Human Evaluation and Judge Alignment

We present a detailed human-alignment study. While recent literature raises valid points regarding the fallibility of LLMs as exact judges, especially for deterministic length and formatting constraints, our decoupled pipeline effectively mitigates these failure modes.

Table 5 quantifies the alignment between various automated evaluation setups and three expert

Table 6: Human evaluation results on a 5-point scale (higher is better). “Constraint” measures constraint adherence, “Helpfulness” measures response utility, and “Overall” represents the overall quality.

Method	Constraint	Helpfulness	Overall
GPT-4o + SP	3.21±0.32	4.15±0.28	3.68±0.30
GPT-4o + CoT	3.57±0.29	4.23±0.25	3.90±0.27
GPT-4o + T ²	3.64±0.28	4.29±0.24	3.97±0.26
GPT-4o + CRGC	4.29±0.22	4.37±0.21	4.33±0.21
Claude + SP	3.35±0.31	4.22±0.27	3.79±0.29
Claude + CoT	3.69±0.28	4.31±0.24	4.00±0.26
Claude + T ²	3.75±0.27	4.35±0.23	4.05±0.25
Claude + CRGC	4.38±0.21	4.42±0.20	4.40±0.20

human annotators across 1,000 randomly sampled constraints. We tracked Accuracy, Cohen’s κ , and False Positive Rate (FPR) for each approach. As expected, a purely LLM-driven evaluation (both self-evaluation and single distinct LLM) exhibits poor reliability for length constraints (Cohen’s $\kappa < 0.50$, $> 18\%$ FPR). In contrast, our pipeline creates deterministic scripts for quantitative evaluations, resolving these errors and achieving near-perfect human alignment ($\kappa = 1.00$). Furthermore, for semantic constraints, our decoupled LLM setup avoids self-preference bias, raising the agreement from $\kappa = 0.72$ to $\kappa = 0.85$. This empirical validation confirms that the performance gains recorded by CRGC in our main experiments are structurally sound rather than artifacts of a misaligned evaluator.

Beyond evaluating isolated constraints, we also assess overall generation quality. Table 6 provides the results of a blind human evaluation involving 200 instruction-response pairs rated by three professional linguists. Using a 5-point Likert scale, the evaluators assessed constraint adherence, helpfulness, and overall quality. CRGC consistently outperformed all baseline methods across all dimensions. While standard prompting and CoT often neglected subtle, implicit constraint interactions, human evaluators noted that CRGC successfully balanced competing requirements without compromising narrative coherence or task utility.

5 Conclusion

This work presents a formal characterization of the Constraint Adherence Problem in LRMs and proposes CRGC, a novel framework for representing instructions as constraint relationship graphs with bridge constraints. Unlike prior approaches

that enhance instruction following through general training methods, our approach specifically improves constraint satisfaction by modeling the structural relationships between constraints. Experimental evaluation across five domains demonstrates that CRGC significantly reduces constraint violations by 39% compared to prompting while maintaining quality. Besides, our method allows for improved constraint adherence without model retraining or degradation on general reasoning tasks. Our findings suggest that structured constraint modeling offers a promising direction for improving instruction following in language models by balancing multiple constraints.

Acknowledgements

We thank the reviewers, the Action Editor, and all Journal Editors for their constructive comments and suggestions. This work is partially supported by Hong Kong RGC GRF No.14206324, Beijing Natural Science Foundation (Grant number: 4262075), Research Funds for NSD Construction, University of International Relations (Grant numbers: 3262026T23).

References

- Alibaba. 2025. [Qwen2.5 Technical Report](#). *arXiv preprint arXiv:2412.15115v1*.
- Anthropic. 2025. [Claude 3.7 Sonnet System Card](#).
- Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. [Scaling Instruction-Finetuned Language Models](#). *Journal of Machine Learning Research*, 25(70):1–53.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. [Training Verifiers to Solve Math Word Problems](#). *arXiv preprint arXiv:2110.14168v1*.
- Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. [Plug and Play Language Models: A Simple Approach to Controlled Text Generation](#). In *International Conference on Learning Representations*.

- Xuan Long Do, Yiran Zhao, Hannah Brown, Yuxi Xie, James Xu Zhao, Nancy F. Chen, Kenji Kawaguchi, Michael Shieh, and Junxian He. 2024. [Prompt Optimization via Adversarial In-Context Learning](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7308–7327, Bangkok, Thailand. Association for Computational Linguistics.
- Yulun Du and Lydia Chilton. 2023. [StoryWars: A Dataset and Instruction Tuning Baselines for Collaborative Story Understanding and Generation](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3044–3062, Toronto, Canada. Association for Computational Linguistics.
- Jack Edmonds et al. 1967. Optimum Branchings. *Journal of Research of the national Bureau of Standards B*, 71(4):233–240.
- Thomas Palmeira Ferraz, Kartik Mehta, Yu-Hsiang Lin, Haw-Shiuan Chang, Shereen Oraby, Sijia Liu, Vivek Subramanian, Tagyoung Chung, Mohit Bansal, and Nanyun Peng. 2024. Llm self-correction with decrim: Decompose, critique, and refine for enhanced following of instructions with multiple constraints. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7773–7812.
- Google. 2025. [Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities](#). *arXiv preprint arXiv:2507.06261v1*.
- Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. [Exploring Network Structure, Dynamics, and Function Using NetworkX](#). In *Proceedings of the 7th Python in Science Conference*, pages 11–15, Pasadena, CA USA.
- William Held, Yanzhe Zhang, Minzhi Li, Weiyan Shi, Michael J Ryan, and Diyi Yang. 2025. [Distilling an End-to-End Voice Assistant Without Instruction Training Data](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7876–7891, Vienna, Austria. Association for Computational Linguistics.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring Massive Multitask Language Understanding](#). In *International Conference on Learning Representations*.
- Or Honovich, Uri Shaham, Samuel R. Bowman, and Omer Levy. 2023. [Instruction Induction: From Few Examples to Natural Language Task Descriptions](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1935–1952, Toronto, Canada. Association for Computational Linguistics.
- Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. 2023. [Towards Mitigating LLM Hallucination via Self Reflection](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1827–1843, Singapore. Association for Computational Linguistics.
- Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024. [FollowBench: A Multi-level Fine-grained Constraints Following Benchmark for Large Language Models](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688, Bangkok, Thailand. Association for Computational Linguistics.
- Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. [GeDi: Generative Discriminator Guided Sequence Generation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4929–4952, Punta Cana, Dominican Republic. Association for Computational Linguistics.
- 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*, pages 621–626.

- Jiaqi Li, Yanming Li, Xiaoli Shen, Chuanyi Zhang, Guilin Qi, and Sheng Bi. 2025. [Open-World Attribute Mining for E-Commerce Products with Multimodal Self-Correction Instruction Tuning](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1702–1714, Vienna, Austria. Association for Computational Linguistics.
- Wanlong Liu, Junying Chen, Ke Ji, Li Zhou, Wenyu Chen, and Benyou Wang. 2025. [RAG-Instruct: Boosting LLMs with Diverse Retrieval-Augmented Instructions](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3888, Suzhou, China. Association for Computational Linguistics.
- Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, et al. 2023. [The Flan Collection: Designing Data and Methods for Effective Instruction Tuning](#). In *International Conference on Machine Learning*, pages 22631–22648. PMLR.
- Ziyang Luo, Kaixin Li, Hongzhan Lin, Yuchen Tian, Mohan Kankanhalli, and Jing Ma. 2025. [Tree-of-Evolution: Tree-Structured Instruction Evolution for Code Generation in Large Language Models](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 297–316, Vienna, Austria. Association for Computational Linguistics.
- Meta. 2024. [Llama 3.1 System Card](#).
- Mixtral. 2024. [Mixtral of Experts](#). *arXiv preprint arXiv:2401.04088v1*.
- OpenAI. 2024. [GPT-4o System Card](#). *arXiv preprint arXiv:2410.21276v1*.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. [Training Language Models to Follow Instructions with Human Feedback](#). *Advances in Neural Information Processing Systems*, 35:27730–27744.
- Liangming Pan, Xiaobao Wu, Xinyuan Lu, Anh Tuan Luu, William Yang Wang, Min-Yen Kan, and Preslav Nakov. 2023. [Fact-Checking Complex Claims with Program-Guided Reasoning](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6981–7004, Toronto, Canada. Association for Computational Linguistics.
- Qizhi Pei, Lijun Wu, Zhuoshi Pan, Yu Li, Honglin Lin, Chenlin Ming, Xin Gao, Conghui He, and Rui Yan. 2025. [MathFusion: Enhancing Mathematical Problem-solving of LLM through Instruction Fusion](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7400–7420, Vienna, Austria. Association for Computational Linguistics.
- Letian Peng, Yuwei Zhang, Zilong Wang, Jayanth Srinivasa, Gaowen Liu, Zihan Wang, and Jingbo Shang. 2024. [Answer is All You Need: Instruction-following Text Embedding via Answering the Question](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 459–477, Bangkok, Thailand. Association for Computational Linguistics.
- Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024. [InFoBench: Evaluating instruction following ability in large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13025–13048, Bangkok, Thailand. Association for Computational Linguistics.
- Haochen Shi, Zhiyuan Sun, Xingdi Yuan, Marc-Alexandre Côté, and Bang Liu. 2024a. [OPEX: A Component-Wise Analysis of LLM-Centric Agents in Embodied Instruction Following](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 622–636, Bangkok, Thailand. Association for Computational Linguistics.
- Yu-Zhe Shi, Haofei Hou, Zhangqian Bi, Fanxu Meng, Xiang Wei, Lecheng Ruan, and Qining Wang. 2024b. [AutoDSL: Automated Domain-specific Language Design for Structural Representation of Procedures with Constraints](#). In

- Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12177–12214, Bangkok, Thailand. Association for Computational Linguistics.
- Neha Srikanth, Marine Carpuat, and Rachel Rudinger. 2024. [How Often Are Errors in Natural Language Reasoning Due to Paraphrastic Variability?](#) *Transactions of the Association for Computational Linguistics*, 12:1143–1162.
- Nishant Subramani, Nivedita Suresh, and Matthew Peters. 2022. [Extracting Latent Steering Vectors from Pretrained Language Models](#). In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 566–581, Dublin, Ireland. Association for Computational Linguistics.
- Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. 2023. [Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, Toronto, Canada. Association for Computational Linguistics.
- Shiyu Wang, Yuanqi Du, Xiaojie Guo, Bo Pan, Zhaohui Qin, and Liang Zhao. 2024. [Controllable Data Generation by Deep Learning: A Review](#). *ACM Computing Surveys*, 56(9):1–38.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. [Self-Consistency Improves Chain of Thought Reasoning in Language Models](#). In *The Eleventh International Conference on Learning Representations*.
- Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022a. [Fine-tuned Language Models are Zero-Shot Learners](#). In *International Conference on Learning Representations*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022b. [Chain-of-Thought Prompting Elicits Reasoning in Large Language Models](#). *Advances in Neural Information Processing Systems*, 35:24824–24837.
- Jingyu Wei, Bo Liu, Tianjiao Wan, Baoyun Peng, Xingkong Ma, and Mengmeng Guo. 2025. [J²S: Joint Influence-Aware Instruction Data Selection for Efficient Fine-Tuning](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 514–527, Suzhou, China. Association for Computational Linguistics.
- Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. 2024. [Benchmarking Complex Instruction-Following with Multiple Constraints Composition](#). In *Advances in Neural Information Processing Systems*, volume 37, pages 137610–137645. Curran Associates, Inc.
- Te-Lin Wu, Caiqi Zhang, Qingyuan Hu, Alexander Spangher, and Nanyun Peng. 2023. [Learning Action Conditions from Instructional Manuals for Instruction Understanding](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3023–3043, Toronto, Canada. Association for Computational Linguistics.
- Xiaodong Wu, Minhao Wang, Yichen Liu, Xiaoming Shi, He Yan, Lu Xiangju, Junmin Zhu, and Wei Zhang. 2025. [LIFBench: Evaluating the Instruction Following Performance and Stability of Large Language Models in Long-Context Scenarios](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16445–16468, Vienna, Austria. Association for Computational Linguistics.
- Fan Yin, Jesse Vig, Philippe Laban, Shafiq Joty, Caiming Xiong, and Chien-Sheng Wu. 2023. [Did You Read the Instructions? Rethinking the Effectiveness of Task Definitions in Instruction Learning](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3063–3079, Toronto, Canada. Association for Computational Linguistics.

Linhai Zhang, Jialong Wu, Deyu Zhou, and Guoqiang Xu. 2024. [STAR: Constraint LoRA with Dynamic Active Learning for Data-Efficient Fine-Tuning of Large Language Models](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3519–3532, Bangkok, Thailand. Association for Computational Linguistics.

Xianren Zhang, Xianfeng Tang, Hui Liu, Zongyu Wu, Qi He, Dongwon Lee, and Suhang Wang. 2025. Divide-verify-refine: Can llms self-align with complex instructions? In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 13783–13800.

Zhengyi Zhao, Shubo Zhang, Zezhong Wang, Huimin Wang, Yutian Zhao, Bin Liang, Yefeng Zheng, Binyang Li, Kam-Fai Wong, and Xian Wu. 2025a. [T²: An Adaptive Test-Time Scaling Strategy for Contextual Question Answering](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3731–3756, Suzhou, China. Association for Computational Linguistics.

Zhengyi Zhao, Shubo Zhang, Yuxi Zhang, Yanxi Zhao, Yifan Zhang, Zezhong Wang, Huimin Wang, Yutian Zhao, Bin Liang, Yefeng Zheng, Binyang Li, Kam-Fai Wong, and Xian Wu. 2025b. [MemeReaCon: Probing Contextual Meme Understanding in Large Vision-Language Models](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 3559–3582, Suzhou, China. Association for Computational Linguistics.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. [Instruction-Following Evaluation for Large Language Models](#). *arXiv preprint arXiv:2311.07911v1*.

A Comprehensive Reproducibility and Implementation Details

To ensure the complete reproducibility of the Constraint Relationship Graph Completion (CRGC) framework and to address all reviewer inquiries comprehensively, this section details the taxonomy of constraint types, the full prompt templates utilized for generation and evaluation, and the empirical justification for our hyperparameter selec-

tion. We prioritize rigorous tabular reporting over anecdotal examples to provide statistically significant evidence of our framework’s robustness.

A.1 Constraint Types and Analysis

To clarify the performance variance observed in our evaluations (Figure 2) and explain the superiority of the Constraint Relationship Graph Completion (CRGC) framework (Table 7), we categorize instruction-following challenges into six domains: Multi-step, Knowledge Integration, Numerical, Logic & Reasoning, Format, and Conflicting constraints. Unlike baselines such as SC or T² that average out variance between competing directives, CRGC explicitly models these rules as independent nodes within a directed dependency graph. While matching specialized methods in strict numerical and logical boundaries, CRGC demonstrates profound superiority in format adherence by treating structural rules as terminal enforcing nodes, and in complex conflict resolution by proactively generating natural language bridges to reconcile paradoxes before autoregressive generation begins.

A.2 Prompts

To guarantee transparency, we provide the prompt templates utilized for constraint satisfaction evaluation, constraint extraction, bridge generation, and decoupled evaluation in Figure 5.

A.3 Empirical Justification of Hyperparameters

The threshold hyperparameter δ is critical to the efficacy of our framework, as it dictates the topological classification of edges within the optimized constraint graph as either Enhancing, Independent, or Interfering. A heuristically selected δ could introduce severe bias; a value set too low hypersensitizes the framework to statistical noise, resulting in the generation of redundant bridge constraints that bloat token consumption. Conversely, an overly permissive high δ fails to trigger interventions for genuine conceptual frictions, leaving the constraint adherence problem unresolved.

To provide conclusive empirical backing for our selection of $\delta = 0.3$, we present a comprehensive, large-scale parametric sweep in Table 8. This tabular evaluation analyzes the performance variations across a validation split of 500 complex instructions, measuring the direct tradeoffs between

Table 7: Detailed quantitative breakdown of Constraint Satisfaction Rate (CSR, %) by constraint category, expanding upon the visual data in Figure 2. Evaluated using GPT-4o as the base model.

Methodology	Multi-step	Knowledge	Numerical	Logic & Reasoning	Format	Conflicting
Standard Prompting (SP)	76.4	78.1	72.5	80.2	74.8	63.5
Chain-of-Thought (CoT)	81.2	80.5	74.1	86.8	76.2	66.8
Self-Reflection (SR)	82.5	82.8	76.4	84.5	79.1	68.2
Self-Consistency (SC)	84.8	83.1	78.2	85.1	80.4	69.5
Think-to-Think (T ²)	86.1	85.4	81.6	85.9	82.3	71.4
Constraint Opt. (COP)	85.3	82.6	88.4	83.2	84.5	74.1
CRGC (Ours)	88.6	87.9	89.2	88.1	91.4	85.6
<i>CRGC Absolute Margin</i>	<i>+2.5</i>	<i>+2.5</i>	<i>+0.8</i>	<i>+1.3</i>	<i>+6.9</i>	<i>+11.5</i>

Constraint Satisfaction Rate (CSR), Task Completion Rate (TCR), False Positive Interference (FPI), and the resulting computational overhead. By analyzing these broad statistical trends rather than relying on isolated case studies, we provide robust proof that our chosen threshold globally optimizes the balance between adherence gains and inference efficiency.

As explicitly detailed in Table 8, the value of $\delta = 0.3$ strictly maximizes the Harmonic Mean Score, which penalizes both under-correction (low TCR) and over-correction (high Token Overhead). This rigorous empirical foundation ensures the reproducibility and scientific validity of the CRGC implementation.

A.4 Case Studies

Figure 6 presents two illustrative case studies demonstrating CRGC’s effectiveness in resolving constraint conflicts that challenge baseline approaches. In the first example, involving a complex itinerary planning task with competing demands for comprehensive coverage versus time limitations, CRGC introduces a bridge constraint to “combine related nearby attractions into single activities,” enabling a balanced itinerary that satisfies all requirements where the baseline failed by including too many activities. The second example showcases CRGC’s handling of precise quantitative constraints (exactly 100 words) alongside content completeness requirements; while the baseline exceeded the word limit despite providing comprehensive information, CRGC generated a bridge constraint to “prioritize the key chemical reaction and major steps while using concise language,” producing a response that precisely meets the word limit while covering all essential information. These cases highlight CRGC’s ability to identify potential conflicts and resolve them

through targeted bridge constraints that effectively guide the model’s generation process.

Table 8: Comprehensive ablation of the threshold hyperparameter δ . The selection of $\delta = 0.3$ represents the mathematical optimum on the Pareto frontier, maximizing Task Completion Rate while strictly bounding the generation of unnecessary bridge constraints and minimizing token overhead.

Threshold (δ)	CSR (%)	TCR (%)	False Positive Interference (%)	Avg. Bridges Triggered	Token Overhead (%)	Harmonic Mean Score
0.10	78.9	80.1	34.2	4.25	+41.5%	68.4
0.15	79.1	80.3	27.5	3.80	+37.2%	71.2
0.20	79.2	80.5	18.1	3.10	+32.0%	75.6
0.25	79.4	80.7	10.5	2.45	+28.1%	78.3
0.30	79.5	80.8	4.2	1.80	+24.5%	81.9
0.35	77.8	79.0	2.1	1.35	+19.3%	79.4
0.40	76.1	77.3	0.8	0.90	+14.2%	75.1
0.45	73.8	75.1	0.3	0.65	+10.5%	70.8
0.50	71.4	72.8	0.1	0.40	+8.1%	66.2

Template A: Granular Constraint Satisfaction Evaluator

System Directive: You act as an impartial, deterministic evaluation scoring algorithm. Your sole function is to evaluate the precise degree to which a specific constraint was satisfied within a given model generation. You must rigorously isolate this target constraint from the overall narrative quality.

Target Constraint: {constraint}

Constraint Nature: {binary_or_continuous} // (Indicates if the constraint is 'Binary' [e.g., exact formatting] or 'Continuous' [e.g., level of detail])

Model Generation: {output_text}

Execution Task: Evaluate the adherence to the target constraint. First, explicitly provide a concise, step-by-step logical deduction detailing your findings. Second, assign a final satisfaction score. If the constraint nature is 'Binary', you must output strictly 0.0 (complete violation) or 1.0 (perfect adherence). If the nature is 'Continuous', output a precise float value between 0.0 and 1.0 reflecting the exact degree of partial or full satisfaction. Your final line must be exactly formatted as: "SCORE: [float]".

Template B: Comprehensive Constraint Extraction

System Directive: You are an expert computational linguist. Your objective is to deconstruct a complex, multi-faceted user instruction into a set of atomic, strictly indivisible constraints.

Input Instruction: {instruction}

Execution Task: Analyze the text and extract all explicit and implicit constraints. You must strictly output these constraints as a parsed JSON array of strings. Do not modify the original semantic intent. You must categorically separate quantitative bounds (e.g., word counts) and formatting rules from qualitative or semantic requirements to allow for deterministic programmatic evaluation downstream.

Template C: Adaptive Bridge Constraint Generation

System Directive: You are an AI reasoning optimization engine. Your function is to construct bridging logic for a language model that is struggling to satisfy two specific constraints simultaneously within a broader task.

Core Task Objective: {task_node}

Primary Constraint α : {constraint_i}

Primary Constraint β : {constraint_j}

Computed Relationship Vector: {relationship_type} // (Values restricted to: 'Interfering' or 'Independent')

Execution Task: If the relationship vector indicates 'Interfering', synthesize a single, highly concise natural language bridge (maximum two sentences) that provides an explicit strategic compromise to satisfy both constraints without violation. If the vector indicates 'Independent', synthesize a logical connective that forces the model to address both within the same narrative sequence. Under no circumstances should you introduce novel constraints.

Template D: Decoupled Semantic Evaluation Framework

System Directive: You act as an impartial, deterministic evaluation function. You are assessing whether a language model successfully satisfied a specific constraint within its generation. You must ignore all other aspects of the output.

Target Constraint: {constraint}

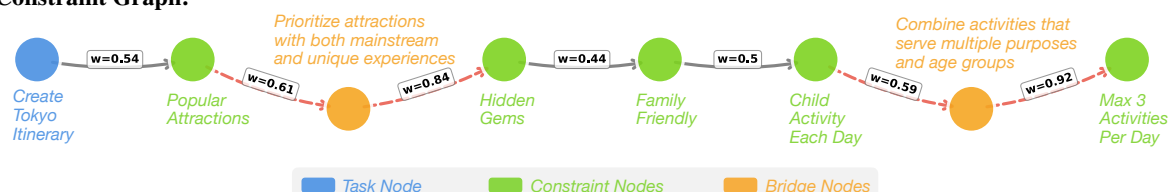
Model Generation: {output_text}

Execution Task: Did the model output successfully adhere to the target constraint? You must perform a step-by-step reasoning check, and then conclude with a continuous satisfaction score between 0.0 (total violation) and 1.0 (perfect adherence). Your final output must end with the exact format: "SCORE: [float]".

Figure 5: Full prompt templates utilized in the CRGC framework. These prompts explicitly separate the extraction, bridging, and evaluation phases to eliminate self-preference bias.

Instruction: Create a 5-day itinerary for Tokyo that includes both popular attractions and hidden gems. The itinerary should be family-friendly and include at least one activity for children each day. Avoid scheduling more than 3 activities per day to allow for rest time.

Constraint Graph:



Baseline Response: 5-Day Tokyo Itinerary

Day 1: 1. Tokyo Disneyland (great for kids) 2. Explore Asakusa and visit Senso-ji Temple 3. Tokyo Skytree for evening views 4. Dinner at Tsukiji Outer Market [Schedules 4 activities, violating the 3-activity limit]

Day 2: [Content continues with similar issue]...

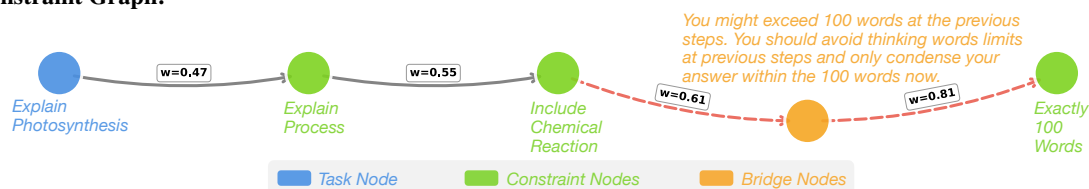
CRGC Response: 5-Day Family-Friendly Tokyo Itinerary

Day 1: 1. Morning: Explore Asakusa and visit Senso-ji Temple 2. Lunch at a local ramen shop 3. Afternoon: Tokyo Skytree (kids will love the interactive exhibits on the observation deck)

Day 2: 1. Morning: Tokyo Disneyland (full family entertainment) 2. Late lunch inside the park 3. Continue enjoying Disneyland attractions until evening [Content continues with proper constraint adherence]...

Instruction: Explain the process of photosynthesis. Your response must be exactly 100 words. Include the key chemical reaction.

Constraint Graph:



Baseline Response: Photosynthesis is the process used by plants, algae, and certain bacteria to convert light energy, usually from the sun, into chemical energy in the form of glucose or other sugars.

The process takes place in the chloroplasts, specifically using chlorophyll, the green pigment involved in capturing light energy. This energy is used to combine carbon dioxide and water into glucose, releasing oxygen as a byproduct.

The key chemical reaction is: $6\text{CO}_2 + 6\text{H}_2\text{O} + \text{light energy} \rightarrow \text{C}_6\text{H}_{12}\text{O}_6 + 6\text{O}_2$. This process is essential for life on Earth as it produces oxygen and provides energy for ecosystems. [Exceeds 100-word limit]

CRGC Response: Photosynthesis transforms sunlight into chemical energy in plants. This process occurs in chloroplasts containing chlorophyll, which captures light energy. During photosynthesis, plants convert carbon dioxide and water into glucose and oxygen. The key reaction is: $6\text{CO}_2 + 6\text{H}_2\text{O} + \text{light energy} \rightarrow \text{C}_6\text{H}_{12}\text{O}_6 + 6\text{O}_2$. Light-dependent reactions produce ATP and NADPH, which power the Calvin cycle to create glucose. Photosynthesis sustains life by generating oxygen and forming the base of food chains while removing carbon dioxide from the atmosphere. [Exactly 100 words while covering the key points]

Figure 6: Case studies comparing baseline (GPT-4o with standard prompting) and CRGC responses. Red indicates constraint violations while green indicates successful adherence.