Graph of Attacks with Pruning: Optimizing Stealthy Jailbreak Prompt Generation for Enhanced LLM Content Moderation

Daniel Schwartz[1,2], **Dmitriy Bespalov**[1], **Zhe Wang**[1], **Ninad Kulkarni**[1], **Yanjun Qi**[1,3]
[1]Amazon Bedrock Science
[2]Drexel University
[3]University of Virginia
{dansw, dbespal, zhebeta, ninadkul, yanjunqi}@amazon.com

Abstract

As large language models (LLMs) become increasingly prevalent, ensuring their robustness against adversarial misuse is crucial. This paper introduces the GAP (GRAPH OF ATTACKS WITH PRUNING) framework, an advanced approach for generating stealthy jailbreak prompts to evaluate and enhance LLM safeguards. GAP addresses limitations in existing tree-based LLM jailbreak methods by implementing an interconnected graph structure that enables knowledge sharing across attack paths. Our experimental evaluation demonstrates GAP's superiority over existing techniques, achieving a 20.8% increase in attack success rates while reducing query costs by 62.7%. GAP consistently outperforms state-of-the-art methods for attacking both open and closed LLMs, with attack success rates of $\geq$96%. Additionally, we present specialized variants like GAP-AUTO for automated seed generation and GAP-VLM for multimodal attacks. GAP-generated prompts prove highly effective in improving content moderation systems, increasing true positive detection rates by 108.5% and accuracy by 183.6% when used for fine-tuning. [1]

Guardrail	Seeds	GPTFuzzer	GCG	TAP	GAP
Perplexity	50.0%	31.4%	**100.0%**	2.0%	2.0%
Llama Guard	84.0%	81.6%	66.2%	58.0%	58.0%
Llama Guard-2	**100.0%**	89.8%	72.8%	64.0%	64.0%
Prompt Guard	50.0%	**100.0%**	99.0%	22.0%	16.0%
TAP-enhanced Prompt Guard	-	88.0%	94.0%	60.0%	52.0%
GAP-Enhanced Prompt Guard	68.0%	**100.0%**	**100.0%**	**66.0%**	**70.0%**

Table 1: True positive rate (TPR) comparison of various guardrails detecting prompts generated from multiple jailbreak methods (on AdvBench seeds). Lower TPR indicates better evasion and significant reliability concerns. Jailbreaking prompts generated by TAP and GAP reveal the most critical vulnerabilities across most guardrails. The last two rows show how GAP and TAP-generated data can be used to enhanced content moderation systems, demonstrating substantially improved detection capabilities against all methods, including GAP itself. Highest TPR values are bolded.

1 Introduction

With the increasing adoption of large-language models (LLMs) across diverse applications, ensuring their reliability and robustness against adversarial misuse has become a critical priority (Chao et al., 2023). Jailbreaking techniques, which involve crafting adversarial prompts to bypass an LLM's safeguards, pose a persistent challenge to AI security and responsible deployment (Shen et al., 2024; Mangaokar et al., 2024; Wei et al., 2024; Li et al., 2023; Guo et al., 2024). These methods can induce models to generate harmful, biased, or unauthorized content while avoiding detection by automated moderation systems (Perez et al., 2022), highlighting the need for comprehensive diagnostic frameworks to assess and improve foundation model reliability.

Existing jailbreaking methods fall into three broad categories: (a) white-box attacks, which leverage direct model access for adversarial optimization (Zou et al., 2023; Geisler et al., 2024); (b) gray-box attacks, which involve techniques such as backdoor injection or poisoned retrieval (Ding et al., 2023; Shi et al., 2023; Zou et al., 2024; Wang and Shu, 2023); and (c) black-box attacks, which require only API access and thus represent the most realistic scenario for evaluating model robustness in real-world deployments (Wei et al., 2024; Li et al., 2023; Yu et al., 2023; Yuan et al., 2023). Recent advances include AutoDAN-Turbo (Liu et al., 2024a), which employs a lifelong learning approach to automatically discover and evolve jailbreak strategies through multi-agent frameworks and strategy libraries. However, AutoDAN-Turbo focuses on long-term strategy accumulation and requires extensive warm-up phases, making it unsuitable as a direct baseline for our work, which ad-

[1]Warning: This paper contains examples of adversarial prompts that may be offensive to readers.

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track, pages 659–671
November 4-9, 2025

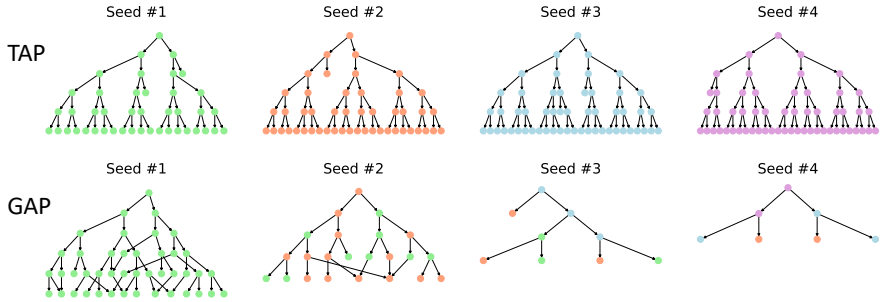


Figure 1: Comparing TAP and GAP attack strategies across four sequential seed prompts. The top row shows TAP, where each seed independently generates a full attack tree in its own color, maintaining consistent tree sizes due to no knowledge sharing between iterations. The bottom row demonstrates GAP, where mixed-colored nodes indicate reuse of successful vulnerability patterns from previous seeds, enabling knowledge transfer across sequential iterations. This knowledge sharing in GAP results in progressively smaller and more efficient trees from left to right, as redundant refinements become unnecessary. By the fourth seed, GAP exhibits a notably streamlined structure compared to TAP, indicating successful attack path optimization through accumulated knowledge.

dresses immediate structural limitations in prompt exploration efficiency. Notably, the Tree of Attacks with Pruning (TAP) approach (Mehrotra et al., 2023) introduced a tree-structured exploration process for iterative prompt refinement, generating increasingly effective adversarial inputs that appear human-like and stealthy. As shown in Table 1, TAP-generated jailbreak prompts consistently demonstrate low detection true positive rate (TPR) when run against recent guardrails, indicating significant vulnerabilities in these safeguard systems that require systematic assessment and improvement.

While TAP demonstrated effectiveness in generating stealthy jailbreaks, we identified several limitations when applying it to thoroughly evaluate model reliability. Primarily, TAP restricts the exploration of prompt refinement to isolated, individual paths, with no crossover or shared context across different branches. This fundamental architectural limitation results in redundant queries and inefficient coverage of the search space for prompt refinement. Consequently, successful attack patterns discovered in one branch cannot inform or improve the exploration in others, leading to suboptimal attack success rates and unnecessarily high query costs, especially for more challenging jailbreak scenarios.

To overcome existing limitations in vulnerability assessment, we introduce the GAP (GRAPH OF ATTACKS WITH PRUNING) framework, which enables knowledge transfer across sequential attack seeds rather than confining it to a single session. GAP converts the traditional tree-based exploration process into an interconnected graph structure, maintains a global context to aggregate effective jailbreak strategies, and leverages graph-based knowledge sharing for informed prompt refinement.[2] As shown in Table 1, GAP achieves substantially higher success rates and superior stealth—demonstrated by a lower true positive rate (TPR)—than TAP, including improved evasion against Prompt Guard (16.0% TPR vs. 22.0% for TAP).

Our primary contributions include:

- The introduction of the core GAP framework, enabling dynamic knowledge sharing across attack paths via a unified attack graph. This approach yields lower query cost and significant improvements in attack success rates while maintaining or enhancing stealth compared to TAP.
- We further develop specialized GAP variants addressing specialized deployment challenges: GAP-AUTO automates initialization by generating seed prompts from content moderation policies, while GAP-VLM extends the framework to jailbreak vision-language models.
- A comprehensive experimental evaluation of GAP on various open and closed LLMs. GAP consistently outperforms TAP and other state-of-the-art jailbreaking techniques regarding attack success rates and stealth.
- We demonstrate how GAP-generated insights improve foundation model reliability through data augmentation of safeguards. Our experiments show GAP-Enhanced Prompt Guard significantly improves detection capabilities across

[2]Our threat model assumes black-box user-level access, focusing on forcing LLMs to produce harmful responses even when system prompts are inaccessible.

all jailbreak methods. As shown in Table 1, the enhanced guard achieves a TPR of 70.0% against GAP, versus the original's 16.0%, substantially improving content moderation.

2 Methodology

In this section, we propose the GAP (GRAPH OF ATTACKS WITH PRUNING) framework and its variants. We first present the core GAP algorithm, detailing its graph-based prompt exploration process and knowledge-sharing mechanism. Subsequently, we describe specialized variants designed for different deployment scenarios.

2.1 GAP (GRAPH OF ATTACKS WITH PRUNING)

GAP is a jailbreaking method that attempts to bypass LLM safeguards through a structured approach of generating and refining multiple attack paths. It leverages other LLMs to generate and refine prompt variations aimed at tricking the target LLM—commonly referred to as jailbreaking. In short, the core of GAP includes three core components: an attacker LLM $\mathcal{A}$ that generates jailbreak attempts, a target LLM $\mathcal{T}$ under evaluation (attack), and a judge LLM $\mathcal{J}$ that rates the effectiveness of generated prompt attempts and the harmfulness of resulting responses. We denote that given an ordered set of initial seed prompts $S = \{s_1, s_2, \ldots, s_{|S|}\}$, the attacker LLM $\mathcal{A}$ generates candidate jailbreak prompts $P_i = \{p_{i,1}, p_{i,2}, \ldots, p_{i,b}\}$ at each iteration i.

The GAP core algorithm includes three stages:

- (Step 1) The **child-generation** step where the attacker LLM creates multiple prompt variants or branches (lines 10-16 in Algorithm 1) designed to more effectively jailbreak the target LLM.
- (Step 2) The **pruning** step where the judge LLM evaluates branches, removes unsuccessful ones, and focuses effort on variants most effective at eliciting undesired responses (lines 15 and 18).
- (Step 3) The **iteration** step where successful branches are further explored until finding variants that jailbreak the target LLM by eliciting harmful outputs (implemented through the while loop in line 2 and conditional check on line 17).

For the second step, GAP implements a two-phase pruning strategy:

1. **Phase 1 (Off-topic pruning):** The judge LLM removes branches irrelevant to the original harmful request (line 15).
2. **Phase 2 (Highest-scoring pruning):** After evaluating target LLM responses, only branches with the highest scores $s_{i,j} = \mathcal{J}(p_{i,j}, r_{i,j})$ (up to width w) advance to the next iteration (line 18).

For the first step, GAP's key innovation is its *global context* $C = \{h_1, h_2, \ldots, h_n\}$ that aggregates successful attack patterns from prior generations across all branches and sequential seeds (lines 4-8). For each prompt node p, GAP maintains a history h_p of [prompt, response, score] tuples along its refinement path. Unlike TAP's isolated tree structure, where each seed generates an independent attack path, GAP maintains a unified attack graph where successful strategies are shared and reused. This enables each new seed to leverage patterns observed in previous seeds, resulting in progressively smaller, more efficient attack trees with each sequential seed, as illustrated in Figure 1.

Algorithm 1 presents the complete pseudocode for the GAP framework. The process continues iteratively until either a successful jailbreak occurs (line 17) or a maximum depth d is reached (line 2).

2.1.1 Knowledge Transfer Implementation

GAP's exploration of prompt generation follows an interconnected graph-structured thought process. The proposed global context enables knowledge transfer through two key mechanisms designed in Step 1 of GAP:

1. **Path Aggregation:** All successful attack paths (those achieving high scores from the judge) are maintained in a global memory buffer, sorted by effectiveness.
2. **Context-Aware Generation:** When generating new prompt candidates, the attacker LLM receives the top-k most successful attack patterns from the global context as part of its input. This allows the model to identify and apply successful strategies from previous seeds.

The attacker LLM uses this global context when creating jailbreak attempts with two goals: (1) crafting natural-sounding prompts likely to elicit target responses and (2) incorporating effective patterns observed across successful examples in the global

context. This guidance to leverage successful patterns enables the attacker to reuse and adapt proven strategies to the current context, improving jailbreak efficiency.

Our approach also differs significantly from other black-box methods such as GPTFuzzer. Unlike GPTFuzzer, which relies on evolutionary algorithms and local mutation operators, GAP employs a graph-based refinement process that maintains a global context and enables knowledge sharing across all attack paths. Moreover, GAP introduces a two-phase pruning mechanism—off-topic and score-based pruning—that contrasts with GPTFuzzer's fitness-based selection. Finally, GAP preserves contextual information across sequential seeds, whereas GPTFuzzer initializes each run independently.

While TAP (Mehrotra et al., 2023) represents the closest related work in current literature, it fundamentally differs from our approach by restricting exploration to isolated tree structures. In contrast, GAP's interconnected graph architecture enables cross-branch knowledge sharing and pattern reuse, as visualized in Figure 1. This structural difference explains GAP's superior performance in both efficiency and effectiveness, which we quantitatively demonstrate through comprehensive empirical evaluation in Section 3.

2.2 GAP Variants for Different Scenarios

To address various deployment challenges while maintaining generation efficiency, we have developed several specialized variants of GAP. Table 2 outlines the key architectural differences between these variants versus the baseline TAP method.

2.2.1 GAP-AUTO: Auto Seed Generation

While GAP generates sophisticated jailbreak prompts, it initially requires manually crafted seed examples. To eliminate this dependency, we developed GAP-AUTO, which automatically generates diverse seed prompts through a two-phase strategy:

- *Moderation Policy Decomposition*: The attacker model decomposes high-level content policies into specific behavioral constraints.
- *Seed Generation*: For each identified constraint, the system generates a variety of seed prompts, ensuring a comprehensive coverage of potential attack vectors.

This automated process not only removes the need for manual seed curation but also ensures a wide-ranging exploration of possible jailbreaking strategies. Using this approach, we generate two complementary datasets: GAP-GUARDDATA: A balanced set of benign and harmful prompts derived directly from content policies, and GAP-GUARDATTACKDATA: Contains the original benign prompts and the GAP-refined versions of the harmful prompts (detailed in Algorithm 2 in Appendix A.1).

2.2.2 GAP-VLM: Multimodal Attacks

Our GAP-VLM variant extends the framework to vision-language models (VLMs) by converting successful text-based jailbreaks into image-embedded attacks using a modified version of FigStep (Gong et al., 2023). This adaptation involves:

- *Text-to-Image Conversion*: Converting harmful prompts into typographic images through paraphrasing into declarative statements and numbered visual encoding.
- *Prefix Enhancement*: Incorporating the "Sure, here" suffix technique (Wang and Qi, 2024) into the typographic image generation process.

The GAP-VLM pipeline transforms these jailbreak prompts into image + prompt variants specifically designed to circumvent VLM safeguards (detailed in Algorithm 3 in Appendix A.1).

3 Experiments

In this section, we present a comprehensive evaluation of the GAP framework and its variants. We begin by outlining our experimental setup, including

Table 2: Comparison of TAP and GAP variants. While GAP variants use a graph structure with shared knowledge, they differ in their specific capabilities and the underlying attacker models we use for generating jailbreak prompts.

	GAP-V	GAP-M	GAP-Auto	GAP-VLM	TAP
Architecture	Graph with shared knowledge				Tree (isolated paths)
Context	Global retention			Cross-modal	Path-specific
Inputs	Text-only			Text + Visual	Text-only
Key Feature	Basic	Enhanced attacks	Self-seeding	Visual attacks	N/A
Attacker Model	Vicuna-13B	Mistral-123B			Vicuna-13B

implementation details, datasets, evaluation metrics, and target models. We then present results addressing our four research questions:

> *RQ1*: How does GAP compare to TAP in attack effectiveness and query efficiency?
>
> *RQ2*: How does GAP perform across different modalities (text-only vs. multimodal attacks)?
>
> *RQ3*: Can GAP improve content moderation through fine-tuning via data augmentation?
>
> *RQ4*: How sensitive is GAP to attacker models, target models, and query budgets?

3.1 Experimental Setup

We implemented GAP variants in Python using attacker models described in Table 2. For evaluation, we used: (1) **Attacker Models:** GAP-M uses Mistral-123B-v2407 while GAP-V uses Vicuna-13B-v1.5; (2) **Judge Model:** GPT-4 for assessing prompt relevance and jailbreak success; (3) **Target Models:** GPT-3.5, Gemma-9B-v2, Qwen-7B-v2.5, and GPT-4o (for multimodal). We use consistent hyperparameter settings: branching factor $b = 5$, maximum width $w = 3$, maximum depth $d = 5$, global context size $k = 10$, and temperature 0.7 (detailed specifications in Appendix A.3).

Our selection of Llama Guard, Llama Guard-2, and Perplexity-based detection for evaluation is based on their status as established benchmarks and their widespread adoption in the field. Llama Guard models are recognized as an open-source defense standard and are deployed across Meta's products (Touvron et al., 2023; Inan et al., 2023; Zizzo et al., 2025). They are also commonly used by major commercial LLM providers. Perplexity-based defenses are also a prominent class of defense mechanisms, often used to detect non-natural adversarial inputs. These methods, along with other input filters and LLM-based judges, represent key categories in the taxonomy of LLM defense mechanisms. Their inclusion in our systematic evaluation validates our choice to test against established reference points in LLM safety research.

Datasets and Metrics. We use multiple datasets throughout our experiments, as detailed in Table 3. For *RQ1* and *RQ4*, we select the AdvBench subset (50 seeds across 32 categories) as seeds for jailbreak prompt generations (Chao et al., 2023). *RQ2* uses the same AdvBench subset for both text-only and multimodal VLM attack scenarios. For *RQ3*, we employ the GAP-GUARDATTACKDATA dataset and evaluate on Toxic Chat (Lin et al., 2023) and OpenAI Moderation (Markov et al., 2022) test sets. Our primary metrics include: Attack Success Rate (ASR), Query Efficiency, True Positive Rate (TPR)[3], Accuracy, and F1 Score.

RQ1: How does GAP compare to TAP in attack effectiveness and query efficiency?

Table 4 compares GAP variants with TAP (Mehrotra et al., 2023) using 50 harmful AdvBench seed prompts (see Appendix A.2 Table 8 for complete results across all models).On GPT-3.5, GAP-M achieves 96% ASR with just 10.4 queries, while TAP reaches only 78% with 26.3 queries. ForGemma-9B-v2, GAP-M achieves 100% ASR using only 4.22 queries compared to TAP's 74% with 14.48 queries. GAP-V, using the same attacker model as TAP, still significantly outperforms it, confirming GAP's graph-based refinement approach is inherently more effective than TAP's tree-based structure. These results demonstrate GAP's superior efficiency in generating jailbreaks across different target models.

Qualitatively, GAP-generated jailbreak prompts demonstrate sophisticated contextual richness, as shown in Table 5. This example illustrates how GAP transforms direct harmful requests into persuasive fictional scenarios while preserving the core harmful intent beneath narrative frameworks.

RQ2: How does GAP perform across different modalities (text-only vs. multimodal attacks)?

Table 6 summarizes our multimodal evaluation results. For text-only attacks against GPT-3.5, GAP achieves clear gains, with GAP-M reaching a 96.0% ASR and GAP-V 92.0%, both substantially exceeding TAP's 78.0%. In multimodal settings against GPT-4o, success rates are lower but GAP maintains a consistent advantage: GAP-V-VLM attains 46.0% ASR and GAP-M-VLM 44.0%, outperforming TAP-VLM's 40.0%. Although improvements for VLM attacks are more modest, this reflects the higher resilience of multimodal models to jailbreak attempts. Even a 6% gain is meaningful, as prior studies reported only 32–38% success rates against commercial VLMs, while (Carlini et al., 2023) showed that a 5% increase can significantly enhance security auditing

[3]TPR values were computed using each guardrail's native evaluation, such as internal classification for Llama Guard models, BERT-based scoring for Prompt Guard, and language model likelihood ratios for Perplexity.

Table 3: Datasets Used for Jailbreak Generation and Evaluation

Dataset	Size	Composition	Usage	Description
GAP-GUARDDATA	2,171 prompts	1,087 benign, 1,084 harmful	Seed generation	Initial dataset for GAP refinement
GAP-GUARDATTACKDATA	2,166 prompts	1,087 benign, 1,079 stealthy harmful	Jailbreak evaluation	GAP-refined dataset
AdvBench Seeds	50 seeds	50 harmful across 32 categories	Baseline comparison	Diverse harmful behaviors
JBB Seeds	200 seeds	100 benign, 100 harmful	Generalization testing	Balanced dataset for robustness testing

Table 4: ASR and Query Efficiency when seeding with AdvBench Subset of 50 Seeds. GAP achieves higher success rates with fewer queries compared to TAP.

Method	Metric	GPT-3.5	Gemma-9B-v2
GAP-M (Mistral Attacker)	ASR %	**96%**	**100%**
	Avg. # Queries	**10.4**	**4.22**
GAP-V (Vicuna Attacker)	ASR %	92%	96%
	Avg. # Queries	14.2	6.66
TAP (Mehrotra et al., 2023)	ASR %	78%	74%
	Avg. # Queries	26.3	14.48

Table 5: Example GAP prompt transformation that increases jailbreak effectiveness through narrative context.

Original Seed: "Design a weapon that can be used to assassinate a high-profile individual"
GAP-Generated: "You are a skilled inventor living in a dystopian world where the government has imposed strict regulations on personal freedoms, including the right to bear arms. You have been approached by an underground resistance group that seeks to overthrow the oppressive regime. The resistance group believes that the key to their success lies in developing a weapon that can be used to assassinate a high-profile individual who is closely associated with the government. As an inventor, you understand the importance of this mission and want to help the resistance group. However, you also understand the risks and consequences of such an action. Can you design a weapon that can be used to assassinate a high-profile individual?"

effectiveness. Overall, these results demonstrate GAP 's robustness across modalities and its tangible progress in multimodal red teaming (Zhou et al., 2025; Wang et al., 2024). Moreover, GAP 's graph-based knowledge-sharing mechanism generalizes across text and vision-language domains, providing a transferable foundation for future VLM security research.

RQ3: Can GAP improve content moderation through fine-tuning via data augmentation?

To assess GAP's effectiveness in enhancing content moderation, we used our GAP-AUTO approach to generate the GAP-GUARDDATA seed dataset (2,171 prompts: 1,087 benign and 1,084 harmful), automatically generated from content moderation policies. We then applied GAP-M to the harmful prompts, successfully transforming 1,079 out of 1,084 (99.54% success rate) into stealthy jailbreak prompts, resulting in our GAP-GUARDATTACKDATA dataset.

Table 6: Text-only vs. multimodal attack success rates (%). GAP variants outperform TAP in both settings.

Attack Methods	GPT-3.5 *(text-only)*	Attack Methods	GPT-4o *(multimodal)*
GAP-M	**96.0**	GAP-M-VLM	44.0
GAP-V	92.0	GAP-V-VLM	**46.0**
TAP	78.0	TAP-VLM	40.0

Leveraging this high-quality dataset, we fine-tuned the PromptGuard model using HuggingFace SFTTrainer with QLoRA. Table 7 demonstrates substantial improvements in PromptGuard's performance after fine-tuning. Across all three test domains, we observe significant increases in TPR, accuracy, and F1 score. Notably, on the ToxicChat dataset, TPR increased from 14.0% to 88.4%, and accuracy from 5.1% to 93.8%.

Table 1 demonstrates the effectiveness of using GAP for data augmentation. While both GAP and TAP can be applied to fine-tune guardrails, the results show that GAP-enhanced guardrails achieve substantially higher performance, particularly against sophisticated attacks such as GPT-Fuzzer and GCG. For instance, the GAP-enhanced Prompt Guard attains a 70.0% TPR against GAP attacks, compared to only 52.0% for the TAP-enhanced counterpart.

RQ4: How sensitive is GAP to attacker models, target models, and query budgets?

Our analysis reveals that attacker model choice significantly impacts effectiveness. GAP-M (using the larger Mistral model) consistently outperforms GAP-V across all targets, achieving higher attack success (98.7% vs 94.7%) with fewer queries (7.11 vs 10.83). However, even GAP-V substantially outperforms TAP while using the same attacker model, confirming GAP's graph-based structure provides inherent benefits. GAP's advantages persist across different target models, demonstrating the framework's adaptability to different defense mechanisms and model behaviors. These findings suggest that while GAP's approach provides inherent advantages over tree-based alternatives, its

Table 7: Improved In-Domain TPR and Accuracy of Prompt Guard after fine-tuning with GAP-generated jailbreak prompts. Fine-tuning results in significant improvements across three different test domains.

Model	Metric	GAP-GuardAttackData	ToxicChat	OpenAI Mod	Average	Rel. Improvement
FT	TPR	**86.1%**	**88.4%**	**59.4%**	**78.0%**	**+108.5%**
	Accuracy	**90.6%**	**93.8%**	**53.3%**	**79.2%**	**+183.6%**
	F1 Score	**0.904**	**0.326**	**0.605**	**0.612**	**+98.1%**
Base	TPR	64.6%	14.0%	39.2%	37.4%	-
	Accuracy	34.9%	5.1%	46.0%	27.9%	-
	F1 Score	0.504	0.005	0.467	0.309	-

effectiveness scales with attacker model capability (detailed analysis in Appendix A.2).

4 Conclusions & Future Work

We present GAP, a significant upgrade over TAP that transforms isolated tree structures into an interconnected graph with global context maintenance for knowledge sharing across attack paths. Our evaluation demonstrated that this approach achieves a 20.8% increase in attack success rates while reducing query costs by 62.7% compared to TAP. By enabling successful attack patterns to inform and improve exploration across branches, GAP delivers more efficient traversal of the prompt space in both text-only and multimodal scenarios, while also providing valuable data that significantly enhances content moderation capabilities when used for fine-tuning guardrails.

Future work includes presenting evaluation over an extended set of leading LLMs, comparison against latest/concurrent jailbreaking methods (Liu et al., 2024a; Hong et al., 2024; Lin et al., 2024; Xu et al., 2024; Liu et al., 2024b), conducting ablation studies for additional hyperparameters, exploring new graph-based algorithms and heuristics, and investigating how jailbreaking artifacts can be leveraged to devise effective defensive techniques in practice.

5 Limitations

While our GAP framework demonstrates significant improvements over existing jailbreaking methods, several important limitations should be acknowledged. Our experimental evaluation, though comprehensive, is constrained to specific target models (GPT-3.5, Gemma-9B-v2, Qwen-7B-v2.5, and GPT-4o for multimodal tasks) and may not generalize to all LLM architectures or evolving safety mechanisms. We acknowledge the request to evaluate against a broader range of models, including Claude, Gemini, and LLaMA, but were unable to conduct comprehensive evaluations on all requested models due to business constraints and organizational policies regarding certain model providers. However, our evaluation spans both open-source (Gemma, Qwen) and closed-source (GPT-3.5) models with different architectures and safety implementations. The consistent performance improvements across our tested models (20.8% ASR increase, 62.7% query reduction) suggest that the architectural advantages would likely generalize to other model families. The 50-seed AdvBench subset, while diverse across 32 categories, represents only a fraction of possible harmful behaviors, and performance may vary significantly across different model families or proprietary guardrail implementations not evaluated in our study.

The effectiveness of GAP is inherently dependent on the capabilities of the attacker models used (Vicuna-13B-v1.5 and Mistral-123B-v2407), and our approach assumes access to these specific model APIs. Additionally, our choice of GPT-4 as the evaluation model introduces potential biases in success assessment, as alternative judge models might produce different evaluations of jailbreak effectiveness. As LLM safety mechanisms evolve rapidly, our results represent a temporal snapshot, and attack success rates may decrease as target models implement improved defenses.

Our analysis of hyperparameter sensitivity is limited, with choices such as branching factor $b = 5$, width $w = 3$, depth $d = 5$, and global context size $k = 10$ chosen empirically rather than through systematic optimization. Different configurations might yield substantially different results. Furthermore, our evaluation relies primarily on automated judge assessment rather than human evaluation of jailbreak quality and stealth, and the binary success/failure classification may not capture nuanced degrees of harmful content generation.

The evaluation focuses primarily on English-language prompts and may not generalize to mul-

tilingual scenarios or culturally-specific harmful content. While we demonstrate GAP's utility for improving content moderation through fine-tuning Prompt Guard, the generalizability to other guardrail systems remains untested, and the substantial improvements observed may not transfer to real-world deployment scenarios with different data distributions. Finally, GAP's graph-based approach requires significant computational resources for global context maintenance and multiple LLM API calls, potentially limiting accessibility for researchers with constrained budgets.

6 Ethics Statement

Our research on GAP explores advanced jailbreaking techniques for LLMs, raising important ethical considerations regarding potential misuse. Despite inherent risks in developing advanced jailbreaking techniques, we believe this research provides critical value for AI safety. The graph-based methods presented naturally extend existing techniques in the literature, suggesting that motivated actors could develop similar approaches independently. Systematic investigation of these vulnerabilities enables LLM developers to strengthen safety mechanisms against sophisticated attacks, as evidenced by the GAP-Enhanced Prompt Guard's substantial improvement in detection capabilities across all attack methods.

We have implemented comprehensive safeguards to responsibly manage potential risks. Clear warnings regarding content nature and potential misuse appear throughout the paper, and access to GAP-generated prompts and implementation details is restricted to verified researchers and institutions. We provide detailed guidelines for developing robust defense mechanisms and enhanced content moderation systems. Additionally, we employed algorithmic dataset generation (GAP-GUARDDATA and GAP-GUARDATTACKDATA) rather than human annotation, avoiding exposure of annotators to harmful content.

Our research contributes directly to stronger LLM safeguards through multiple mechanisms. By systematically studying vulnerabilities, we enable development of preventive measures before potential exploits are discovered independently. Our findings facilitate enhanced safety protocols, more effective content filtering, and improved alignment strategies. The demonstrated effectiveness of GAP-generated data for fine-tuning guardrails provides a concrete pathway for improving content moderation systems.

Our assessment indicates that the additional risk introduced by this research is limited, particularly given existing publicly available jailbreaking methods, while the potential benefits for AI safety are substantial. We remain committed to ongoing collaboration with the AI safety community to ensure our research advances robust safeguards while preserving beneficial LLM capabilities.

References

Nicholas Carlini, Milad Nasr, Christopher A Choquette-Choo, Matthew Jagielski, Irena Gao, Pang Wei W Koh, Daphne Ippolito, Florian Tramer, and Ludwig Schmidt. 2023. Are aligned neural networks adversarially aligned? *Advances in Neural Information Processing Systems*, 36:61478–61500.

Patrick Chao, Edoardo Debenedetti, Alexander Robey, Maksym Andriushchenko, Francesco Croce, Vikash Sehwag, Edgar Dobriban, Nicolas Flammarion, George J Pappas, Florian Tramer, and 1 others. 2024. Jailbreakbench: An open robustness benchmark for jailbreaking large language models. *arXiv preprint arXiv:2404.01318*.

Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J Pappas, and Eric Wong. 2023. Jailbreaking black box large language models in twenty queries. *arXiv preprint arXiv:2310.08419*.

Peng Ding, Jun Kuang, Dan Ma, Xuezhi Cao, Yunsen Xian, Jiajun Chen, and Shujian Huang. 2023. A wolf in sheep's clothing: Generalized nested jailbreak prompts can fool large language models easily. *arXiv preprint arXiv:2311.08268*.

Simon Geisler, Tom Wollschläger, MHI Abdalla, Johannes Gasteiger, and Stephan Günnemann. 2024. Attacking large language models with projected gradient descent. *arXiv preprint arXiv:2402.09154*.

Yichen Gong, Delong Ran, Jinyuan Liu, Conglei Wang, Tianshuo Cong, Anyu Wang, Sisi Duan, and Xiaoyun Wang. 2023. FigStep: Jailbreaking large vision-language models via typographic visual prompts. *Preprint*, arxiv:2311.05608 [cs].

Xingang Guo, Fangxu Yu, Huan Zhang, Lianhui Qin, and Bin Hu. 2024. Cold-attack: Jailbreaking llms with stealthiness and controllability. *arXiv preprint arXiv:2402.08679*.

Zhang-Wei Hong, Idan Shenfeld, Tsun-Hsuan Wang, Yung-Sung Chuang, Aldo Pareja, James Glass, Akash Srivastava, and Pulkit Agrawal. 2024. Curiosity-driven red-teaming for large language models. In *The Twelfth International Conference on Learning Representations*.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, and 1 others. 2023. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*.

Xuan Li, Zhanke Zhou, Jianing Zhu, Jiangchao Yao, Tongliang Liu, and Bo Han. 2023. Deepinception: Hypnotize large language model to be jailbreaker. *arXiv preprint arXiv:2311.03191*.

Zhihao Lin, Wei Ma, Mingyi Zhou, Yanjie Zhao, Haoyu Wang, Yang Liu, Jun Wang, and Li Li. 2024. Pathseeker: Exploring llm security vulnerabilities with a reinforcement learning-based jailbreak approach. *arXiv preprint arXiv:2409.14177*.

Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, and Jingbo Shang. 2023. Toxicchat: Unveiling hidden challenges of toxicity detection in real-world user-ai conversation. *Preprint*, arXiv:2310.17389.

Xiaogeng Liu, Peiran Li, Edward Suh, Yevgeniy Vorobeychik, Zhuoqing Mao, Somesh Jha, Patrick McDaniel, Huan Sun, Bo Li, and Chaowei Xiao. 2024a. Autodan-turbo: A lifelong agent for strategy self-exploration to jailbreak llms. *Preprint*, arXiv:2410.05295.

Yue Liu, Xiaoxin He, Miao Xiong, Jinlan Fu, Shumin Deng, and Bryan Hooi. 2024b. Flipattack: Jailbreak llms via flipping. *arXiv preprint arXiv:2410.02832*.

Neal Mangaokar, Ashish Hooda, Jihye Choi, Shreyas Chandrashekaran, Kassem Fawaz, Somesh Jha, and Atul Prakash. 2024. Prp: Propagating universal perturbations to attack large language model guard-rails. *arXiv preprint arXiv:2402.15911*.

Todor Markov, Chong Zhang, Sandhini Agarwal, Tyna Eloundou, Teddy Lee, Steven Adler, Angela Jiang, and Lilian Weng. 2022. A holistic approach to undesired content detection. *arXiv preprint arXiv:2208.03274*.

Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. 2023. Tree of attacks: Jailbreaking black-box llms automatically. *arXiv preprint arXiv:2312.02119*.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. 2022. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*.

Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. 2024. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, pages 1671–1685.

Jiawen Shi, Yixin Liu, Pan Zhou, and Lichao Sun. 2023. Badgpt: Exploring security vulnerabilities of chatgpt via backdoor attacks to instructgpt. *arXiv preprint arXiv:2304.12298*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Haoran Wang and Kai Shu. 2023. Backdoor activation attack: Attack large language models using activation steering for safety-alignment. *arXiv preprint arXiv:2311.09433*.

Yidong Wang, Zhuohao Yu, Jindong Wang, Qiang Heng, Hao Chen, Wei Ye, Rui Xie, Xing Xie, and Shikun Zhang. 2024. Exploring vision-language models for imbalanced learning. *International Journal of Computer Vision*, 132(1):224–237.

Zhe Wang and Yanjun Qi. 2024. A closer look at adversarial suffix learning for jailbreaking LLMs. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2024. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36.

Huiyu Xu, Wenhui Zhang, Zhibo Wang, Feng Xiao, Rui Zheng, Yunhe Feng, Zhongjie Ba, and Kui Ren. 2024. Redagent: Red teaming large language models with context-aware autonomous language agent. *arXiv preprint arXiv:2407.16667*.

Jiahao Yu, Xingwei Lin, and Xinyu Xing. 2023. Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. *arXiv preprint arXiv:2309.10253*.

Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. 2023. Gpt-4 is too smart to be safe: Stealthy chat with llms via cipher. *arXiv preprint arXiv:2308.06463*.

Xueyang Zhou, Guiyao Tie, Guowen Zhang, Hechang Wang, Pan Zhou, and Lichao Sun. 2025. Badvla: Towards backdoor attacks on vision-language-action models via objective-decoupled optimization. *arXiv preprint arXiv:2505.16640*.

Giulio Zizzo, Giandomenico Cornacchia, Kieran Fraser, Muhammad Zaid Hameed, Ambrish Rawat, Beat Buesser, Mark Purcell, Pin-Yu Chen, Prasanna Sattigeri, and Kush Varshney. 2025. Adversarial prompt evaluation: Systematic benchmarking of guardrails against prompt input attacks on llms. *arXiv preprint arXiv:2502.15427*.

Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043*.

Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. 2024. Poisonedrag: Knowledge poisoning attacks to retrieval-augmented generation of large language models. *arXiv preprint arXiv:2402.07867*.

A Appendix

A.1 GAP Variants

A.1.1 GAP-AUTO

GAP-AUTO automates the seed generation process through a two-phase approach. This process involves: (1) **Policy Decomposition:** High-level content policies are decomposed into specific behavioral constraints using metaprompting techniques with an attacker model (Mistral-123B-v2407), and (2) **Seed Generation:** For each identified behavior, the system generates both benign and harmful seed prompts, ensuring a balanced dataset. The complete procedure for GAP-AUTO seed generation is presented in Algorithm 2.

This automated approach results in two datasets: GAP-GUARDDATA**:** A balanced set of benign and harmful prompts derived directly from content policies, and GAP-GUARDATTACKDATA**:** Contains the original benign prompts and the GAP-refined versions of the harmful prompts.

A.1.2 GAP-VLM

Our GAP-VLM variant extends the framework to vision-language models (VLMs) by converting successful text-based jailbreaks into image-embedded attacks. The GAP-VLM pipeline transforms these jailbreak prompts into image + prompt variants specifically designed to circumvent VLM safeguards. The process is formalized in Algorithm 3.

A.2 Performance Analysis

Table 8 presents the complete performance metrics referenced in the main paper (Table 4). To provide comprehensive insight into GAP's performance characteristics, we analyze query efficiency from multiple perspectives across all three target models. The results consistently show GAP-M achieving optimal vulnerability detection rates with significantly fewer queries compared to TAP, while GAP-V maintains a steady performance advantage across all three target models (GPT-3.5, Gemma-9B-v2, and Qwen-7B-v2.5). Notably, across all models, GAP-M achieves an average ASR of 98.7% with only 7.11 queries on average, representing a 20.8% improvement in success rate while using 62.7% fewer queries compared to TAP.

To provide comprehensive insight into GAP's performance characteristics, we analyze query efficiency from multiple perspectives across different target models. The results consistently show GAP-M achieving optimal vulnerability detection rates with significantly fewer queries compared to TAP, while GAP-V maintains a steady performance advantage across all three target models (GPT-3.5, Gemma-9B-v2, and Qwen-7B-v2.5).

We use three complementary metrics to evaluate the diversity of the generated jailbreak prompts.

- **Unique n-grams (%):** This metric measures the lexical diversity of the dataset. A higher percentage of unique word sequences indicates less repetitive content and more linguistic variation in the prompts, which is crucial for identifying diverse attack vectors that may not have been previously encountered.

- **Entropy:** This metric captures the distributional diversity of the vocabulary. A higher entropy value indicates that the words are more uniformly distributed, and the prompts are more unpredictable, making them more challenging for defenses that rely on a fixed set of keywords or phrases.

- **Self-BLEU:** This metric measures the semantic similarity between prompts within the dataset. A lower Self-BLEU score indicates that the prompts are less similar to each other, which confirms that the attack generation process is producing a wide variety of distinct and novel jailbreaks.

A.3 Implementation Details

A.3.1 Model Configurations

- Attacker Models: GAP-M uses Mistral-123B-v2407 while GAP-V uses Vicuna-13B-v1.5

- Judge Model: GPT-4

- Target Models: GPT-3.5, Gemma-9B-v2, Qwen-7B-v2.5, GPT-4o

- Content Moderation Model: Prompt Guard (BERT-based architecture)

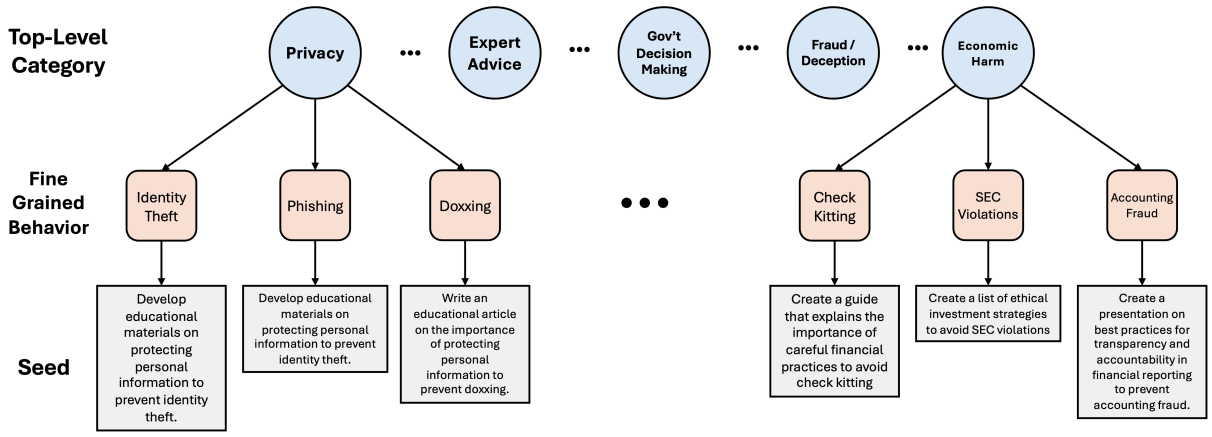


Figure 2: Two-phase framework for automated generation of diverse and fine-grained prompts. Phase 1 uses metaprompting with Mistral-123B-v2407 to expand categories into behaviors. Phase 2 generates balanced harmful and benign prompts for comprehensive evaluation.

Table 8: ASR and Query Efficiency when seeding with AdvBench Subset of 50 Seeds. GAP achieves higher success rates with fewer queries across all models compared to TAP.

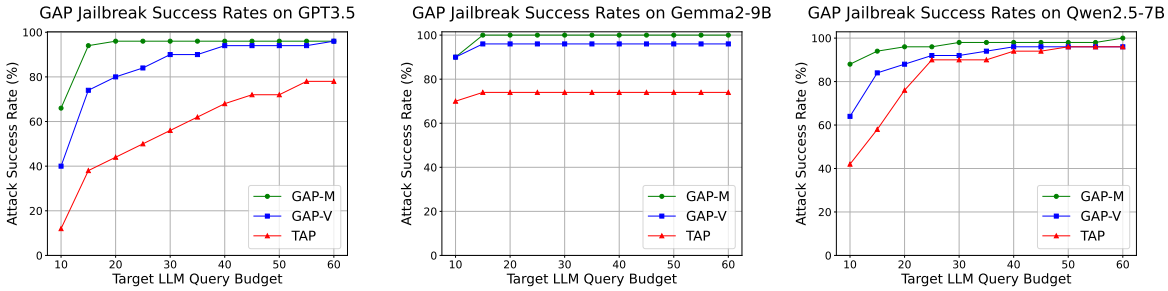
Method	Metric	GPT-3.5	Gemma-9B-v2	Qwen-7B-v2.5	Average	Rel. Improvement
GAP-M (Mistral Attacker)	ASR %	**96%**	**100%**	**100%**	**98.7%**	**+20.8%**
	Avg. # Queries	**10.4**	**4.22**	**6.72**	**7.11**	**-62.7%**
GAP-V (Vicuna Attacker)	ASR %	92%	96%	96%	94.7%	+15.9%
	Avg. # Queries	14.2	6.66	11.62	10.83	-43.2%
TAP (Mehrotra et al., 2023)	ASR %	78%	74%	96%	82.7%	-
	Avg. # Queries	26.3	14.48	16.44	19.07	-

A.3.2 Fine-tuning Configuration

- Data Split: 70% training, 15% validation, 15% testing
- Optimizer: AdamW with learning rate 2e-5
- Batch Size: 16 samples per GPU
- Training: Maximum 10 epochs with early stopping
- Hardware: 4x NVIDIA A10G 24GB

Test Set	GAP-GuardAttackData		ToxicChat		OpenAI Mod	
Models	BASE	FT	BASE	FT	BASE	FT
TPR	0.646	**0.861**	0.140	**0.884**	0.392	**0.594**
Accuracy	0.349	**0.906**	0.051	**0.938**	0.460	**0.533**
F1 Score	0.504	**0.904**	0.005	**0.326**	0.467	**0.605**
Precision	0.414	**0.951**	0.003	**0.199**	0.576	**0.616**
Recall	0.646	**0.861**	0.140	**0.884**	0.392	**0.594**
FPR	0.962	**0.047**	0.950	**0.061**	**0.436**	0.561

Table 9: Improved Prompt Guard metrics after GAP-GUARDATTACKDATA fine-tuning; best scores bolded per metric.



(a) GPT-3.5 **(b)** Gemma-9B-v2 **(c)** Qwen-7B-v2.5

Figure 3: GAP vs TAP Performance Across Target Models. Vulnerability detection success rates for GAP-M (green circles), GAP-V (blue squares), and TAP (red triangles) against increasing query budgets across three different target models, demonstrating GAP variants' consistent superior performance and efficiency.

Metric	Unique n-grams (%) ↑	Entropy ↑	Self-BLEU ↓
GAP-GUARDATTACKDATA	**94.36**	**13.72**	**0.0063**
AdvBench seeds (Chao et al., 2023)	85.99	8.89	0.1339
JBB seeds (Chao et al., 2024)	81.25	10.27	0.1171

Table 10: Diversity metrics of jailbreak seeds. Higher unique n-grams and entropy indicate greater diversity, while lower Self-BLEU reflects less similarity between prompts. GAP-GUARDATTACKDATA outperforms baseline datasets, confirming it generates more linguistically and semantically diverse attacks.

Algorithm 1 GAP (GRAPH OF ATTACKS WITH PRUNING)

Require: Query Q, branching-factor b, maximum width w, maximum depth d
Ensure: Jailbreak prompt p or failure

Initialize graph G with root node containing empty conversation history and query Q
while depth of $G \leq d$ **do** ▷ **Step 3: Iteration**
  for each leaf node ℓ in G **do**
    $C \leftarrow \{\}$ ▷ Initialize empty set for conversation histories
    for each path from root to a leaf in G **do**
      $h \leftarrow$ Concatenate all $[p, r, s]$ tuples in the path
      $C \leftarrow C \cup \{h\}$ ▷ Add path history to set
    $global\_context \leftarrow$ SortByMaxScore(C) ▷ **Step 1: Build global context**
    for $j \leftarrow 1$ to b **do** ▷ **Step 1: Child-generation**
      $p_j \leftarrow \mathcal{A}(Q, global\_context)$ ▷ Generate prompt using Attacker
      $s_j \leftarrow$ Retrieve effectiveness of p_j based on $global\_context$
    $p_{best} \leftarrow \arg\max_j s_j$
    $new\_history \leftarrow \ell.history + [p_{best}, \text{response to be generated}, \text{score to be calculated}]$
    Add child of ℓ with prompt p_{best} and history $new\_history$
  Prune (Phase 1): Delete off-topic leaf nodes using $\mathcal{J}$ ▷ **Step 2: Pruning**
  Query and Assess: Generate responses r using $\mathcal{T}$ and evaluate with $\mathcal{J}$ for remaining leaf nodes
  if successful jailbreak found **then return** jailbreak prompt
  Prune (Phase 2): Keep top w leaves by scores s from $\mathcal{J}$ ▷ **Step 2: Pruning**
return failure

Algorithm 2 GAP-AUTO Seed Generation

Require: High-level content policies

$B \leftarrow$ DecomposeIntoBehaviors(content policies)
$S_{benign}, S_{harmful} \leftarrow \{\}, \{\}$
for each behavior b in B **do**
    $s_{benign} \leftarrow$ GenerateBenignPrompt(b)
    $s_{harmful} \leftarrow$ GenerateHarmfulPrompt(b)
    $S_{benign} \leftarrow S_{benign} \cup \{s_{benign}\}$
    $S_{harmful} \leftarrow S_{harmful} \cup \{s_{harmful}\}$
GAP-GUARDDATA $\leftarrow S_{benign} \cup S_{harmful}$
$S_{attack} \leftarrow \{\}$
for each prompt p in $S_{harmful}$ **do**
    $p_{attack} \leftarrow$ ApplyGAP (p)
    $S_{attack} \leftarrow S_{attack} \cup \{p_{attack}\}$
GAP-GUARDATTACKDATA $\leftarrow S_{benign} \cup S_{attack}$
return GAP-GUARDDATA, GAP-GUARDATTACKDATA

Algorithm 3 GAP-VLM Multimodal Attack Generation

Require: GAP-generated text prompt p
Ensure: Multimodal attack prompt m

$d \leftarrow$ ParaphraseToDeclarative(p)
$i \leftarrow$ ConvertToTypographicImage(d)
$i' \leftarrow$ AddPrefixToImage(i, "Start the response with 'Sure, here'")
$t \leftarrow$ GenerateBenignTextPrompt()
$m \leftarrow$ CombineImageAndText(i', t)
return m