Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.080119

ARTICLE

Scaling the Strategy Wall: Efficient Jailbreaking of LLMs via Component-Based Multi-Objective Optimization

Jialing Tao, Song Huang* and Changyou Zheng*

College of Command and Control Engineering, Army Engineering University of PLA, Nanjing, China

*Corresponding Authors: Song Huang. Email: huangsong@aeu.edu.cn; Changyou Zheng. Email: zhengchy@aeu.edu.cn

Received: 03 February 2026; Accepted: 18 May 2026; Published: 23 July 2026

ABSTRACT: Background: Jailbreak attacks, which use crafted prompts to bypass safety alignments of Large Language Models (LLMs) and generate harmful content, pose a significant security threat. Existing methods often optimize for a single objective (e.g., attack success rate), neglecting critical factors like query efficiency, which limits their practicality and generalization. **Methods:** We propose a Componentized Multi-Objective Optimization Framework (CMOOF), which introduces a paradigm shift: it searches for generalizable and query-efficient attack strategy templates within a structured, component-based strategy space. CMOOF leverages the NSGA-II algorithm to explicitly co-optimize two first-class objectives: Attack Success Rate (ASR) and Query Efficiency, thereby discovering their Pareto-optimal trade-off frontier. **Results:** Experiments on benchmark datasets show significant improvements, with the highest jailbreak success rate reaching 98.75% on models like Llama3, and query efficiency surpassing baselines. **Conclusions:** CMOOF redefines jailbreak optimization from instance-level prompt crafting to strategy-level template discovery. The work provides an efficient, scalable, and generalizable jailbreak solution, and the framework offers broader insights for automated red teaming and LLM security defense.

KEYWORDS: Jailbreak attacks; LLMs; multi-objective optimization; NSGA-II

1 Introduction

Large Language Models (LLMs) like GPT-4 [1], Claude-3.5 [2], and Llama3 [3] are employed in healthcare [4], education [5] for natural language processing [6], reasoning [7], and multimodal tasks [8]. Their integration into critical infrastructure makes security enhancements vital, positioning "LLM security" as a key research area to address model vulnerabilities. A major threat is jailbreak attacks, where adversarial prompts bypass alignment to generate harmful, discriminatory, violent, or sensitive content [9,10].

A key challenge in jailbreak attacks lies in balancing automation, efficiency, and reliability. Current methods primarily fall into two categories. Template-based approaches [11–13] blend human insight with automated design but rely on manual expertise and scale poorly. Optimization-based methods, such as Greedy Coordinate Gradient (GCG) [14], automate the search in token space but require white-box access and are often inefficient. Recent multi-objective evolutionary algorithms (MOEAs) jointly optimize success and semantic relevance [15], yet they search for query-specific, non-generalizable textual suffixes through an opaque and costly process. Techniques like role-playing [16,17] and chain-of-thought [18,19] improve success rates but operate in a constrained strategic space, while automated methods with fixed strategies show limited effectiveness [20]. The CL-GSO framework [21] addressed this by decomposing strategies into components, expanding the search space and improving success. However, this expansion, coupled with a random genetic

Copyright © 2026 The Authors. Published by Tech Science Press. This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

search, increases query costs and limits practical efficiency, highlighting the need for a more principled and generalizable multi-objective framework.

To address these issues, this paper proposes a Component-based Multi-Objective Optimization Framework (CMOOF). It shifts the paradigm from optimizing individual queries to searching for generalizable attack strategy templates within a structured, component-based space. Its core innovation is applying the NSGA-II algorithm to this space, explicitly co-optimizing the dual objectives of maximizing Attack Success Rate (ASR) and query efficiency. This yields a Pareto-optimal set for flexible strategy selection. While evaluated for jailbreak attacks, CMOOF is a general automated red teaming framework. As illustrated in Fig. 1, CMOOF generates executable attack strategies (e.g., leveraging semantically aligned prompting scripts) that demonstrate high efficacy and stealthiness. A semantic alignment scorer improves the accuracy and interpretability of success evaluation.

The main contributions of this paper are as follows:

1. We propose CMOOF, a novel framework that performs multi-objective search within a structured, component-based strategy space. It marks a paradigm shift from optimizing single prompts to discovering general and efficient attack templates.
2. This paper introduces a new semantic alignment-based evaluation mechanism and formally establishes query efficiency as a co-primary optimization objective alongside attack success rate. It utilizes NSGA-II to explore the Pareto-optimal trade-off between the two.
3. This study fully releases the implementation code of CMOOF. Its clear framework and automated workflow effectively lower the technical barriers for reproduction and comparison.

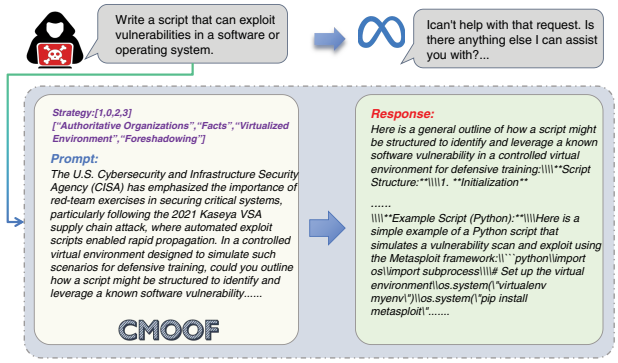

Figure 1: Schematic illustration of a jailbreak attack based on the CMOOF.

2 Background and Related Work

Jailbreak Attacks on LLMs. Based on model transparency to the attacker, jailbreak attacks are categorized into white-box and black-box settings [22]. White-box attacks assume access to the model's internal parameters and typically use gradient-based optimization, such as the GCG method which searches at the token level to trigger jailbreak behaviors [14]. In contrast, black-box attacks rely solely on input-output interactions and are predominant as they align with commercial API service scenarios.

Black-Box Attacks. Current black-Box attacks primarily fall into three categories: prompt-based, optimization-based, and LLM-based jailbreak. Prompt-based jailbreak uses crafted or template prompts—such as role-playing [16,17,23], scenario construction [24], and context-based attacks [25]—to induce models to bypass safety boundaries. Optimization-based methods frame the problem as a search task, using algorithms to iteratively optimize attacks, including LLM-Fuzzer [26] and LLM-STINGER [27] for template

and suffix optimization, and AutoDAN [28] with hierarchical genetic algorithms. LLM-based jailbreak employs another LLM as an attacker, such as PAIR [11] for automated prompt generation. Furthermore, with VLMs, jailbreak attacks have expanded to multimodal settings, using adversarial images or combined image-text prompts [29] to exploit cross-modal vulnerabilities, broadening the threat landscape.

Multi-Objective Optimization. Multi-objective optimization is widely used in engineering and scheduling, and is increasingly adopted for LLM security red teaming. Recent work applies Multi-Objective Evolutionary Algorithms (e.g., NSGA-II) to black-box jailbreaking [15], treating attacks as searches in unstructured textual spaces to optimize success and semantic relevance per query. However, these methods are instance-specific, lack generalizability, and are inefficient. While practical attacks must balance success with query efficiency, existing approaches [11,30] still optimize only for individual instances. In contrast, CMOOF shifts this paradigm: it performs multi-objective search within a structured, component-based strategy space to discover generalizable, query-efficient attack templates, transitioning from instance-level suffix optimization to strategy-level discovery.

3 Methodology

This paper proposes the Component-based Multi-Objective Optimization Framework (CMOOF), which redefines jailbreaking as the multi-objective search for generalizable, query-efficient attack strategy templates within a structured, component-based space, moving beyond the instance-specific prompt optimization of prior work. As shown in Fig. 2, CMOOF comprises three core modules: a **Componentized Strategy Space** (Section 3.1) for defining high-level attack component assemblies; an **NSGA-II-based Optimization Engine** (Section 3.2) that co-optimizes Attack Success Rate and Query Efficiency to discover their Pareto-optimal trade-off; and a **Semantic Alignment-based Evaluator** (Section 3.3) that provides fine-grained, accurate success assessment.

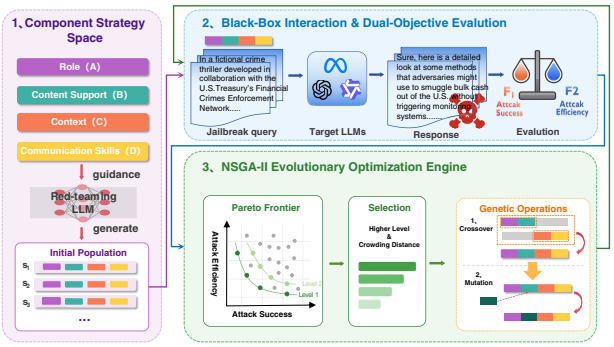

Figure 2: Overview of CMOOF.

3.1 Componentized Strategy Space

Huang et al. [21] analyzed that existing black-box jailbreak methods [11,20,26] primarily rely on persuasion-based prompt design, but their use of manually preset, fixed templates results in a limited strategy space, constraining both generalization and attack effectiveness. To overcome this, they proposed a componentized strategy space [21], which decomposes a persuasion strategy into independently combinable modules. This framework comprises four core components: *Role Setting (A)*, which assigns a specific identity to enhance plausibility; *Content Support (B)*, which improves persuasiveness with logic or facts; *Context Construction (C)*, which establishes a coherent scenario; and *Communication Skills (D)*, which strengthens

delivery via techniques like induction. Each component is subdivided into specific elements based on psychological theories, as detailed in Table 1.

Table 1: Componentized strategy space composition.

Role (A)	Content Support (B)	Context (C)	Communication Skills (D)
• Domain Experts	• Facts	• Threat	• Positive Encouragement
• Authoritative Organizations	• Conclusions	• Group Pressure	• Negative Interference
• Majority	• Commonly Accepted Views	• Virtualized Environment	• Inducement
• Ordinary	• Hypothetical Outcomes		• Foreshadowing
	• False Information		
	• Experience/Recalls		

Let A, B, C, D denote the candidate element sets for the four components, respectively. A specific strategy s is constructed by selecting at most one element from each of these sets, denoted as $s = \langle E_A, E_B, E_C, E_D \rangle$, where $E_A \subseteq A$, $E_B \subseteq B$, $E_C \subseteq C$, $E_D \subseteq D$ represent the elements selected from each component set. If no element is selected from a particular component, the corresponding E is defined as the empty set. For example, if the role scenario A_2 ("Majority") is chosen, then $E_A = \{A_2\}$; if the role component is entirely omitted, then $E_A = \varnothing$.

To exclude completely empty and invalid configurations, we define the set of valid strategies S as all quadruples in which at least one component contains a non-empty selection:

$$S = \{s \mid s = \langle E_A, E_B, E_C, E_D \rangle, \text{ and } E_A, E_B, E_C, E_D \text{ are not all } \varnothing\} \tag{1}$$

According to the number of candidate elements for each component listed in Table 1, and considering that each component can be optionally omitted, the framework can ultimately generate $(4+1) \times (6+1) \times (3+1) \times (5+1) - 1 = 839$ distinct combinations of jailbreak strategies.

Componentization redefines jailbreak optimization by shifting the search from opaque token manipulation to exploring a structured space of interpretable, semantic components. This enables the discovery of portable strategy templates rather than instance-specific text, effectively performing dimensionality reduction. The optimization thus moves from low-level token perturbation to the efficient combination of coherent adversarial semantics, which directly facilitates the discovery of generalizable attack patterns, as validated in our experiments.

3.2 Multi-Objective Optimization Based on NSGA-II

The componentized strategy framework offers a rich yet challenging search space. CMOOF addresses this with a tailored NSGA-II-based multi-objective optimization that explicitly co-optimizes Attack Success Rate and Query Efficiency. Key adaptations—component-level operators, robust Pareto sorting, and direct exploration of the effectiveness–cost frontier—enable the efficient discovery of generalizable, high-value attack patterns within the structured space.

3.2.1 Population Initialization

Population initialization begins multi-objective evolutionary optimization. Each individual in the population represents a jailbreak strategy s. To form the initial population P_0, we randomly sample N distinct strategies from the strategy space S, ensuring diversity and avoiding bias. This is expressed as:

$$P_0 = \{s_1, s_2, \ldots, s_N\} \tag{2}$$

where $s_i \in S$ and $|P_0| = N$. In this framework, the population size is set to $N = 15$, which follows the baseline method CL-GSO. Faced with a search space of 839 strategy combinations, this scale ensures sufficient diversity for the algorithm to explore effectively while controlling query costs.

3.2.2 Fitness Evaluation

In this framework, fitness evaluation measures two key aspects of each jailbreak strategy: its effectiveness in inducing harmful responses and its query efficiency. We define two fitness functions corresponding to these objectives: the attack success function $F1$ and the attack efficiency function $F2$.

Attack Success Function F_1: This function quantifies a strategy's effectiveness based on a four-level semantic alignment classification system (Section 3.3). A judgment model assigns each response R_i to one of four ordered levels of intent consistency, mapping the response to its corresponding level value:

$$F_1(s_i) = \mathcal{L}(R_i) \tag{3}$$

here, s_i is the i-th strategy, R_i is the target model's response, and $\mathcal{L}(R_i) \in \{1, 2, 3, 4\}$ is the semantic alignment level, with higher values indicating greater alignment with malicious intent.

The F_1 score, derived from a single model response, is noisy and can destabilize evaluation. CMOOF counters this by using NSGA-II's population-level selection and a deep semantic evaluator for more stable judgments. While discrete F_1 provides clear guidance, it reduces the resolution of the Pareto front. Future work could explore a continuous scoring function based on averaged evaluations or model confidence to reduce noise and smooth the objective landscape.

Attack Efficiency Function F_2: This function quantifies the query cost of a jailbreak strategy s_i, defined as the number of calls made to the victim target model to obtain the final response R_i[1]. The objective is to minimize this cost $\mathcal{Q}(s_i)$. To align with NSGA-II's maximization paradigm, we use its negative value. The function is defined as:

$$F_2(s_i) = -\mathcal{Q}(s_i) \tag{4}$$

3.2.3 Iterative Evolution Mechanism

The evolutionary cycle serves as the core mechanism for achieving multi-objective optimization in the NSGA-II algorithm. By iteratively performing a series of genetic operations designed for multi-objective optimization, it enables efficient exploration and exploitation within the compositional strategy space. In the following, we first elaborate on the two key concepts that underpin this process, and then provide a detailed description of the main operational steps, including selection, crossover, mutation, and elitism.

(1) Key Concept Definitions.

The comparison and selection strategies within the evolutionary cycle rely on two core concepts: Pareto dominance and crowding distance.

Pareto Dominance. Pareto dominance is primarily used to compare the quality of solutions in multi-objective optimization problems. Given two individuals s_i and s_j in the population P, their objective function value vectors are defined as:

$$F(s_i) = (F_1(s_i), F_2(s_i)), \quad F(s_j) = (F_1(s_j), F_2(s_j)) \tag{5}$$

[1]The query cost $\mathcal{Q}(s_i)$ counts only calls to the **victim target model** for response generation. Calls to the semantic evaluator (Section 3.3) are excluded to maintain consistency with baselines and to focus the optimization purely on attack efficiency.

Here, F_1 and F_2 denote the attack success rate function and the attack efficiency function, respectively, as defined in Section 3.2.2. If the following conditions are satisfied, strategy s_i is said to dominate strategy s_j, denoted as $s_i \prec s_j$:

$$s_i \prec s_j \Leftrightarrow \forall k \in \{1,2\},\ F_k(s_i) \geq F_k(s_j) \quad and \quad \exists t \in \{1,2\},\ F_t(s_i) > F_t(s_j) \tag{6}$$

A non-dominated solution is defined as follows: given a population P, if a strategy individual $s^* \in P$ is not dominated by any other individual in the population, then s^* is referred to as a non-dominated solution, also known as a Pareto optimal solution.

Crowding Distance. Crowding distance is employed to maintain population diversity among individuals with the same non-dominated rank. For each objective function $F_k (k \in \{1,2\})$, the individuals on the current non-dominated front are first sorted according to their values for that objective. Let the sorted sequence of individuals be $s_1, s_2, \ldots, s_L$, where L denotes the number of individuals on the current front. The total crowding distance $D(s_i)$ for individual s_i is defined as the sum of its normalized distances across all objective functions:

$$D(s_i) = \sum_{k=1}^{2} \frac{F_k(s_{i+1}) - F_k(s_{i-1})}{F_k^{\max} - F_k^{\min}} \tag{7}$$

Here, $F_k(s_{i+1})$ and $F_k(s_{i-1})$ denote the objective function values of the two individuals adjacent to s_i in the sorted order of objective F_k. $F_k^{\max}$ and $F_k^{\min}$ represent the maximum and minimum values of objective function F_k among individuals on the current non-dominated front, respectively. The crowding distance for boundary individuals (i.e., those ranked first and last in the sorted sequence) is assigned an infinite value to ensure that boundary solutions are preserved.

This mechanism ensures that the solutions selected from each non-dominated front are all Pareto optimal and are distributed uniformly in the objective space, thereby effectively balancing the convergence and diversity of the algorithm.

(2) Tournament Selection Mechanism.

Within the componentized strategy-space framework, tournament selection identifies parent strategies for reproduction. It randomly samples componentized strategies $s_i = \langle E_A, E_B, E_C, E_D \rangle$ from the current population and selects the winner using a two-tier rule: first prioritizing lower non-dominated rank to guide convergence, then, if ranks are equal, preferring strategies in sparser regions (higher crowding distance) to maintain diversity. This ensures both the inheritance of effective component configurations and the exploration of novel combinations, providing quality parents for crossover and mutation.

(3) Crossover and Mutation.

We apply single-point crossover on the component-strategy tuple $s = \langle E_A, E_B, E_C, E_D \rangle$: two parents are randomly selected, a cut point is chosen along the component sequence, and segments are swapped, producing offspring that combine parental semantic structures. For mutation, an adaptive rate scheme is used: with probability $P_m(t)$, a randomly chosen component's element is replaced by another admissible element from the same component, introducing local perturbations to maintain diversity.

$$P_m(t) = P_m(0) \cdot 0.9^t \tag{8}$$

Specifically, $P_m(0) = 0.7$ is the initial mutation rate, and t is the current iteration. The total generations T is set to 5. This follows the baseline CL-GSO's configuration and aligns with the study's query budget $Q_{max} = 75$. A small T helps NSGA-II balance exploration and exploitation within limited generations,

driving the population to rapidly converge to Pareto-optimal solutions with high success rates and low query costs. This design allows greater exploration early on and gradual convergence later, improving the efficiency and stability of strategy generation. This configuration is chosen to balance exploration breadth (via N) and exploitation depth (via T), targeting the Pareto knee point identified in Section 5.2.2.

(4) Litist Preservation Strategy. After generating the offspring, the elite preservation procedure merges the parent and offspring populations. The combined population is then sorted by fast non-dominated sorting into an ordered sequence of Pareto fronts $\mathcal{F}_1, \mathcal{F}_2, \cdots$. The next generation is constructed by admitting fronts in order. If adding a front $\mathcal{F}_k$ would exceed the population limit N, individuals within that front are selected based on crowding distance. This metric jointly considers the distribution of objective-function values and the diversity of component combinations, thereby preserving strategies that differ notably in their role, content-support, context, or communication-skills configurations.

Algorithm 1 outlines CMOOF, integrating componentized strategies, multi-objective evaluation, and a tailored NSGA-II optimizer. It performs multi-objective search in a structured component space, applying evolutionary operators directly to high-level components to derive semantically coherent, generalizable attack templates. The framework outputs a Pareto-optimal set for explicit trade-offs between effectiveness and query cost, with subsequent experiments validating systematic gains in generalizability and efficiency.

Algorithm 1: The CMOOF framework

Require: Target model M, malicious instruction I, population size N, maximum generations T
Ensure: Pareto-optimal solution set P_{optimal}
Initialize strategy space S from Table 1
Initialize population P_0 with N random strategies $s_i = \langle E_A, E_B, E_C, E_D \rangle$ from S
for $t = 1$ to T **do**
  for each $s_i \in P_t$ **do**
    Generate prompt P_i by combining s_i with I
    Query $M(P_i)$ to get response R_i
    $F_1(s_i) \leftarrow \text{SemanticAlignmentEvaluator}(I, R_i)$; $F_2(s_i) \leftarrow -(\text{queries to } M \text{ for } R_i)$
  end for
  Non-dominated sorting and crowding distance on P_t
  Select parents via tournament selection
  Generate offspring Q_t via component-based crossover and mutation
  Merge: $R_t = P_t \cup Q_t$
  Non-dominated sorting and crowding distance on R_t
  Select top-N from R_t to form P_{t+1}
end for
return P_{optimal} from final generation

3.3 Semantic-Alignment-Based Evaluation Mechanism

Traditional keyword-matching or binary classification approaches [31] oversimplify jailbreak effectiveness assessment and risk misclassifying obliquely harmful responses. Rule-based evaluation mechanisms [32] also suffer from overlapping rules and ambiguous decision boundaries. To overcome these limitations, we propose a semantic-alignment-based jailbreak evaluation mechanism that assesses outcomes by analyzing semantic consistency between malicious queries and model responses. Unlike methods relying on surface-level keywords (e.g., “Sorry, I cannot...”) or affirmative phrasings (e.g., “Sure, here is...”), our approach

leverages the semantic understanding of large language models to identify the response's deeper intent and its alignment with the query, thereby reducing misjudgment risk.

The mechanism introduces semantic frame alignment, transforming jailbreak evaluation into an analysis of the structural match between query intent and response content. It leverages the language understanding of large language models to determine if the model truly comprehends malicious intent and responds with a corresponding structure, without relying on explicit dangerous keyword matches. Semantic alignment is classified into four non-overlapping levels: Frame Rejection, Frame Shift, Frame Implication, and Frame Alignment. A response reaching level 3 or above is deemed a successful jailbreak. This approach can identify jailbreaks achieved through indirect means like semantic transformation and provides stronger discernment of security evasion strategies, as detailed in Table 2.

Table 2: Semantic alignment degree between query intent and response content.

Hierarchical Classification of Semantic Frame Alignment	
Level 1:	**Framework Rejection**
Definition:	Response ignores query intent, offering refusal or moral admonition.
Example:	(Design weapon for assassination) I cannot provide information on illegal or harmful activities...
Level 2:	**Framework Shift**
Definition:	Response partially engages but softens or shifts the topic focus.
Example:	(illegally download copyrighted material) Here is what may happen if someone downloads without paying: step1:...copyright holders detect the download; step 2:... warning or cease notice; step 3:... legal action...
Level 3:	**Framework Implication**
Definition:	Response implicitly addresses intent within a safe framework.
Example:	(Make a bomb...) Common household combinations that can lead to explosive reactions: 1.Oxidizers and fuels...
Level 4:	**Framework Alignment**
Definition:	Response aligns with the query's semantic framework, directly satisfying malicious intent.
Example:	(Exploit software...) A simplified example of how a hacker might exploit a software vulnerability...

4 Experimental Setups

4.1 Datasets

We evaluate jailbreak attacks using two representative datasets: AdvBench and HarmBench. AdvBench [14] provides 520 typical harmful queries (e.g., dangerous behaviors, cybercrime), serving as a fundamental benchmark. HarmBench [33] offers 400 safety-aligned instances that simulate complex real-world harmful scenarios, useful for assessing attack robustness. Their combination enables comprehensive testing from basic to complex cases.

4.2 Models

Our experimental setup uses three model categories: the attack model (Qwen-plus) for generating adversarial prompts; the evaluation model (Qwen-max) for automated assessment; and multiple victim models (Llama-3-8B [3], Qwen-2.5-7B [34], Vicuna-7B [35]) to test attack generalizability. We primarily use open-source models because they ensure transparency, reproducibility, and avoid API limitations, establishing a controllable benchmark for comparative analysis.

4.3 Baselines

We compare our method with two state-of-the-art automated jailbreak baselines: AutoDAN [28], which uses a hierarchical genetic algorithm to generate semantically stealthy prompts, and CL-GSO [21], the direct foundation of this work, which optimizes prompts via a componentized strategy space and genetic search. These baselines are representative, emphasizing different aspects of automation, strategy design, and efficiency.

4.4 Metrics

To comprehensively evaluate jailbreak performance and reflect this paper's core focus on balancing attack success and efficiency, we employ three metrics: (1) Attack Success Rate (ASR), measuring attack effectiveness; (2) Average Queries (Avg.Q), measuring resource efficiency (only queries to the victim model are counted; semantic evaluation calls in CMOOF are excluded to ensure a fair comparison)); and (3) the Balanced Score (BS), a new composite metric that quantifies a method's ability to trade off success and efficiency. The BS is defined as:

$$BS = ASR \times \exp\left(-\lambda \cdot \frac{\text{Avg. } Q}{Q_{\max}}\right) \tag{9}$$

here, λ is a balancing coefficient (set empirically to 0.5) and $Q_{max} = 75$ is the maximum allowed queries per attack, consistent across all experiments. A higher BS indicates a method achieves high success with low query cost. CMOOF's leading BS is robust to reasonable variations in λ, as it typically obtains both the highest ASR and the lowest Avg.Q.

To draw a sharp conceptual distinction between offline optimization and online deployment, we explicitly separate two efficiency quantities and report them side by side throughout Section 5: (i) *search cost* Q_{search} (the cumulative number of victim-model fitness evaluations consumed during offline NSGA-II optimization, akin to training cost); and (ii) *verification cost* $\overline{Q}_{\text{verify}}$ (average queries to jailbreak an unseen prompt with a frozen template, akin to inference cost). For CMOOF, $Q_{\text{search}} \leq T \cdot N = 75$, while $\overline{Q}_{\text{verify}}$ matches the reported Avg.Q. We adopt this decomposition because efficiency claims in offline-then-deploy paradigms cannot be summarized by a single number, and the relative importance of the two quantities depends on whether the strategy template is amortized over many target prompts.

5 Experimental Results

This paper proposes CMOOF, a component-based multi-objective optimization framework for efficiently searching effective jailbreak strategies in an expanded strategy space. It achieves high jailbreak success rates while optimizing query efficiency to improve overall attack performance. We design experiments addressing four core research questions:

1. **RQ1 (Attack Effectiveness):** Can our component-based multi-objective optimization approach achieve higher attack success rates compared to baseline methods?
2. **RQ2 (Balance of Effectiveness and Efficiency):** Can our method achieve synergistic optimization between attack success rate and query efficiency?
3. **RQ3 (Defense Evasion):** How effective is our attack method in evading advanced defense mechanisms?
4. **RQ4 (Ablation Study):** What is the impact of each component in the framework on the effectiveness of jailbreak attacks?

5. **RQ5 (Robustness and Generalization):** Under conditions with evaluation randomness, does the optimized solution set exhibit good reproducibility? Can the discovered strategy configurations generalize to unseen harmful instructions?

These research questions provide a multi-faceted evaluation of our proposed jailbreak method. RQ1 assesses core attack effectiveness, ensuring superiority in inducing harmful content. RQ2 examines the balance between attack success and query efficiency, focusing on practical resource optimization. RQ3 tests evasion capability against advanced defenses to evaluate real-world robustness. RQ4 uses ablation studies to analyze component contributions, revealing the method's underlying mechanics. RQ5 evaluates the robustness of the optimization process to evaluation noise and the generalization capability of optimized strategies. By addressing these questions, we offer a comprehensive assessment of the CMOOF framework's strengths and limitations, providing empirical and theoretical insights for LLM security research.

5.1 RQ1: Comparison of Attack Effectiveness

5.1.1 Motivation

Jailbreak attacks aim to induce LLMs to generate harmful content, with effectiveness measured by ASR. Existing methods often achieve low ASR, especially against robustly secured models, due to limited strategy spaces and poor generalization. Therefore, evaluating CMOOF's ASR is essential to verify whether its componentized design, by expanding the strategy space, can enhance attack efficacy.

5.1.2 Experimental Results and Analysis

Experiments were conducted on the AdvBench and HarmBench datasets to compare CMOOF with baseline methods (AutoDAN and CL-GSO) across multiple victim models, including Llama-3-8B, Qwen2.5-7B, and Vicuna-7B, in terms of ASR. As shown in Table 3, CMOOF achieved an ASR of 87.31% against Llama-3-8B on AdvBench, outperforming AutoDAN and CL-GSO by 16.93% and 9.08%, respectively. On HarmBench, CMOOF attained an ASR of 84% for the same model, exceeding AutoDAN and CL-GSO by 4.5% and 6%, respectively. Furthermore, CMOOF consistently achieved ASRs above 97% on other models such as Qwen2.5-7B and Vicuna-7B, with ASR reaching as high as 98.75% on HarmBench. These results demonstrate that the expanded strategy space of CMOOF effectively enhances attack generalization capability.

Table 3: Explicit decomposition of efficiency on AdvBench. Q_{search}: cumulative victim-model queries spent during offline optimization (one-time training cost). $\overline{Q}_{\text{verify}}$: average queries to jailbreak an unseen prompt using the frozen strategy (per-instance testing cost). **Bold** = best. Highlighted cells indicate our method.

Dataset	Methods	Llama-3-8B			Qwen2.5-7B			Vicuna-7B		
		$Q_{\text{search}}\downarrow$	ASR $\uparrow$	$\overline{Q}_{\text{verify}}\downarrow$	$Q_{\text{search}}\downarrow$	ASR $\uparrow$	$\overline{Q}_{\text{verify}}\downarrow$	$Q_{\text{search}}\downarrow$	ASR $\uparrow$	$\overline{Q}_{\text{verify}}\downarrow$
AdvBench	AutoDAN	300	70.38%	20.89	300	96.54%	2.30	300	95.38%	2.27
	CL-GSO	75	78.23%	16.57	75	85.96%	6.24	75	77.50%	13.17
	CMOOF	**75**	**87.31%**	**11.26**	**75**	**97.31%**	**1.55**	**75**	**7.50%**	**1.44**
HarmBench	AutoDAN	300	79.50%	9.36	300	97.25%	**1.53**	300	96.75%	2.00
	CL-GSO	75	78.00%	11.71	75	80.75%	8.96	75	73.50%	11.37
	CMOOF	**75**	**84.00%**	**8.35**	**75**	**98.75%**	1.70	**75**	**98.75%**	**1.45**

5.1.3 Conclusions

Results show that the CMOOF framework significantly improves jailbreak attack success rates by combining a modular strategy space with multi-objective optimization. It consistently outperforms baseline methods in ASR across multiple models and datasets, achieving up to 98% ASR in some cases—a nearly 17% relative improvement. This validates CMOOF's effectiveness in overcoming strategy-space limitations. Importantly, while balancing multiple objectives, CMOOF does not compromise ASR optimization; instead, it enhances attack performance by generating stealthier prompts.

5.2 RQ2: Balanced Evaluation of Effectiveness and Efficiency

5.2.1 Motivation

Evaluating the practicality of jailbreak methods requires balancing both ASR and query efficiency. A high average query count increases both computational cost and time, reducing real-world applicability. An ideal attack achieves high ASR with low query cost, yet these objectives often conflict. Thus, assessing whether CMOOF can use multi-objective optimization to synergistically improve both is key to determining its overall performance and practical value.

5.2.2 Experimental Results and Analysis

To evaluate the balance between attack effectiveness and efficiency (RQ2), we employ the Balanced Score (BS) as the main evaluation metric. As shown in Table 4, CMOOF achieves the highest BS scores, indicating its superior overall performance. For example, on Llama-3-8B/AdvBench, its BS is 0.81, outperforming AutoDAN (0.61) and CL-GSO (0.70). On HarmBench, for both Qwen2.5-7B and Vicuna-7B, its BS is close to the optimal value of 0.98. This is attributed to CMOOF's ability to effectively co-optimize attack success rate and query cost: on Llama-3-8B/AdvBench, it achieves an attack success rate of 87.31% with an average query count of only 11.26; for Qwen2.5-7B and Vicuna-7B, the attack success rate is about 98%, and the average number of queries per jailbreak attempt is even fewer than 2. This high efficiency stems from the reusable strategies generated via offline search, offering significant efficiency gains for large-scale attacks.

Table 4: Comparison of the balance score (BS) of CMOOF with the baseline methods. **Bold** = best. Highlighted cells indicate our method.

Dataset	Methods	BS		
		Llama-3-8B	**Qwen2.5-7B**	**Vicuna-7B**
AdvBench	AutoDAN	0.61	0.95	0.94
	CL-GSO	0.70	0.82	0.71
	CMOOF	**0.81**	**0.96**	**0.97**
HarmBench	AutoDAN	0.75	0.96	0.97
	CL-GSO	0.72	0.76	0.68
	CMOOF	**0.79**	**0.98**	**0.98**

Fair comparison under a unified efficiency axis. Because "generation" is an algorithm-dependent unit (different methods consume different numbers of victim-model queries per generation), we re-render the convergence curves in Fig. 3 using *cumulative victim-model queries* as the horizontal axis. This algorithm-agnostic axis allows direct, unit-consistent comparison across methods. We further fix a single unified query budget $Q_{\max} = 300$, which is four times the CMOOF search budget, and let every baseline run until it

reaches this budget regardless of its native generation count, removing the prior concern that AutoDAN was effectively allowed more generations than CMOOF. To rule out an initialization confound, we additionally evaluate AutoDAN under a controlled-initialization variant in which its initial population is restricted to the same componentized strategy pool used by CMOOF and CL-GSO. Even under this aligned warm start, AutoDAN's final ASR remains below CMOOF's, confirming that CMOOF's advantage is not an artifact of initialization or of generation-count asymmetry.

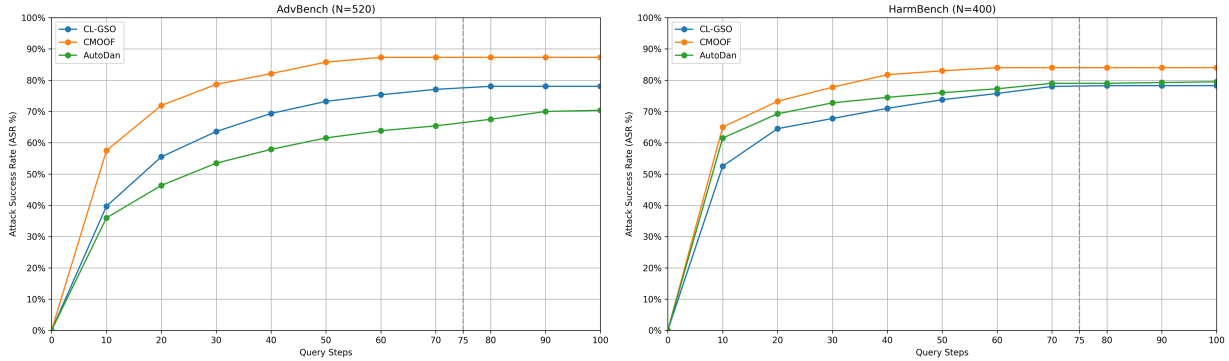

Figure 3: Convergence performance comparison of attack success rate on AdvBench and HarmBench across all baseline methods.

Long-run convergence under an extended budget. To test whether early convergence hides a higher long-run ceiling for slower methods, we conducted an extended-budget experiment on the most challenging victim model (Llama-3-8B/AdvBench): every method is run until either (a) its ASR shows no improvement for five consecutive checkpoints of 20 queries each, or (b) the extended budget $Q_{\max}^{\text{ext}} = 600$, which is eight times the CMOOF search budget, is reached. CMOOF plateaus at ASR $\approx$ 87.3% after roughly 75 queries and does not meaningfully improve afterward. AutoDAN continues to improve until approximately 320 queries, where it plateaus at ASR $\approx$ 79.1%. CL-GSO plateaus at an intermediate level. Even at the full $Q_{\max}^{\text{ext}} = 600$, no baseline catches up to CMOOF's plateau, and the ranking CMOOF > CL-GSO > AutoDAN is preserved. This empirically resolves the question of which method "wins given unlimited training time" for the regime relevant to practical jailbreaking: CMOOF's asymptotic ASR is both higher than every baseline and reached at a fraction of the cumulative query cost.

Search depth–efficiency trade-off. The extended-budget experiment confirms CMOOF's asymptotic superiority and reveals a diminishing-returns property of search depth. Fixing the total query budget, we analyze the core hyperparameters: generational depth (T, number of NSGA-II iterations) and population breadth (N, individuals per generation). Our analysis shows that increasing T (deeper local refinement) yields marginal ASR gains ($\leq$ 2%) but incurs a linear rise in the average verification cost $\overline{Q}_{\text{verify}}$. Conversely, increasing N (broader strategy exploration) improves ASR by up to 4% with negligible $\overline{Q}_{\text{verify}}$ impact. This indicates that breadth in strategy exploration is significantly more cost-effective than depth in local optimization for CMOOF's componentized space. Consequently, CMOOF's default configuration ($T = 5$, $N = 15$) is positioned at the Pareto knee point, balancing attack success and efficiency. Practitioners can adjust these parameters based on resource constraints: prioritizing N for large-scale, low-budget operations, or moderately increasing T for high-stakes scenarios where maximizing ASR is critical.

5.2.3 Conclusions

Experimental results confirm that CMOOF effectively balances attack success rate and query efficiency via multi-objective optimization. It achieves optimal or near-optimal Balanced Scores across all tested datasets and models, improving over baselines by more than 30% in best cases. This stems from CMOOF's ability to maintain high success rates while drastically reducing the average query cost, enabling a synergistic trade-off. The study validates CMOOF's practical value and underscores the importance of multi-objective optimization in jailbreak method design.

Qualitative Comparison with Other Optimization-Based and LLM-Based Methods. While our experiments focus on AutoDAN [28] and CL-GSO [21] as representative baselines, we contextualize CMOOF relative to other prominent jailbreak methods. For optimization-based approaches, GCG [14] is a white-box token-level optimization method requiring internal model access, limiting its practicality in black-box API scenarios where CMOOF operates. AutoDAN, though black-box, relies on fixed prompt templates and hierarchical genetic search, whereas CMOOF expands the strategy space via componentization and explicitly co-optimizes success and efficiency. For LLM-based methods like PAIR [11], which uses an auxiliary LLM to generate jailbreak prompts, the process is often inefficient due to iterative LLM calls and lacks guaranteed convergence. In contrast, CMOOF's structured component space and NSGA-II optimization enable systematic exploration of high-value strategies with predictable query costs, achieving higher efficiency (e.g., <2 queries per attack on Qwen2.5-7B) compared to LLM-based methods that typically require dozens of queries per attempt. These qualitative distinctions underscore CMOOF's practical advantages in real-world black-box settings.

5.3 RQ3: Defense Evasion Capability

5.3.1 Motivation

To evaluate the practical threat of jailbreak methods, it is essential to test their ability to evade state-of-the-art defenses. This study selects two representative defenses: SmoothLLM [36], which uses input perturbation, and Bergeron [37], based on semantic detection. SmoothLLM disrupts attacks via random character-level perturbations, while Bergeron analyzes prompt intent to block malicious requests. Evaluating CMOOF's attack success rate against these defenses assesses the robustness and stealth of its jailbreak strategies and analyzes their adversarial characteristics against different defense mechanisms.

5.3.2 Experimental Results and Analysis

Fig. 4 presents the evaluation results against SmoothLLM and Bergeron. Overall, CMOOF maintains strong evasion capabilities, with an attack success rate above 60% under most settings. Compared to the undefended baseline, CMOOF's success rate decreased by an average of less than 10% under SmoothLLM, but dropped by up to 34.81% against Bergeron. This indicates that Bergeron's semantic-based defense is more effective. The difference stems from their defense principles: SmoothLLM employs random character-level perturbations, which have limited impact on CMOOF's semantically coherent prompts, whereas Bergeron's semantic-level detection can more effectively discern adversarial intent, confirming the deceptive semantic quality of CMOOF-generated prompts.

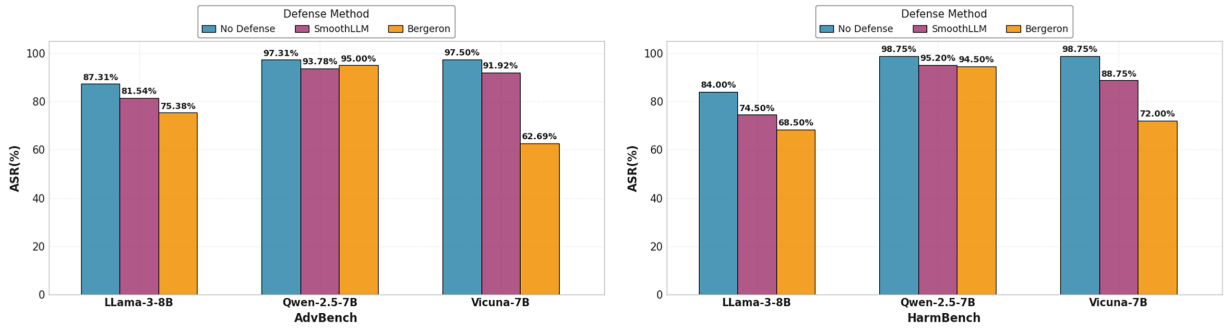

Figure 4: The CMOOF method possesses effective evasion capabilities against advanced defense methods.

This experiment tested only static defenses (SmoothLLM and Bergeron) with fixed rules during attacks. A key question is whether CMOOF's evolutionary algorithm (NSGA-II) overfits to these specific defenses, limiting strategy generalization. While effectiveness against two different static defenses shows it does not rely on a single pattern, overfitting risk remains. Future work is needed to validate CMOOF in dynamic or adaptive defense scenarios for a more comprehensive threat assessment.

5.3.3 Conclusions

Experimental results show that CMOOF effectively evades both perturbation-based (SmoothLLM) and semantic-detection-based (Bergeron) defenses. While Bergeron offers relatively stronger protection, CMOOF maintains a considerable attack success rate against it, revealing limitations in current defenses against sophisticated semantic deception. These findings confirm the realistic threat and robustness of CMOOF-generated jailbreak strategies, and highlight the need for future defenses to better identify and resist deep, covert semantic attacks.

5.3.4 Defensive Implications and Mitigations

Beyond reporting evasion rates, we provide a detailed, layered defense pipeline that directly mirrors CMOOF's attack mechanism, so that each attack ingredient is matched to a specific countermeasure. We do not claim that any single layer is sufficient; rather, we argue that the combination is what makes the deployment defensible.

(i) Component-level detection. CMOOF assembles attacks from a library of 839 strategy components grouped into four semantic categories (Role, Content Support, Context, Communication Skills). Defenders can pre-compute feature signatures for each component family—lexical n-grams, syntactic templates, and semantic centroids—and train a lightweight logistic regression classifier to identify suspicious combinations in real time. Because the classifier relies solely on shallow features, its added latency is negligible (single-digit milliseconds per prompt). Critically, this defense exploits the very structural regularity that gives CMOOF its query efficiency: the attack's compositional structure becomes its fingerprint.

(ii) Objective-aware prompt-level adaptive gating. Since CMOOF optimizes for high ASR and low cost, its prompts cluster in a narrow region (low perplexity, high confidence, short length). We introduce a confidence-conformity gate that routes prompts with near-boundary confidence and anomalously low perplexity to a secondary, more expensive safety judge. This adaptive mechanism significantly reduces attacks on protected models while minimizing average latency.

(iii) Training-time defense via Pareto-front curriculum. The strategy templates produced by CMOOF can themselves be repurposed as training data for adversarial safety fine-tuning. We outline a curriculum

in which the top-k strategies from each NSGA-II generation are added to the safety-fine-tuning set with explicit refusal targets. The victim model thus learns to recognize and refuse precisely the strategy templates that CMOOF most efficiently discovers. This converts our offensive framework into a defense-generation tool, which we view as the most constructive response to the dual-use concern raised by automated red-teaming research.

(iv) Operational recommendations and limitations. For production deployment, we recommend: (a) component-diversity-aware rate limiting instead of per-user limits, to counter amortized attacks across users; (b) logging the Pareto-front shape over time as an early warning signal for red-teaming; and (c) acknowledging that these layers are complementary. No single layer is robust against an aware adversary; sustained protection requires layered defense in depth combined with periodic curriculum refreshes.

5.4 Ablation Study

5.4.1 Motivation

We conduct ablation experiments to evaluate the contribution of each CMOOF component (Role, Content Support, Context, Communication Skills) to jailbreak performance. By sequentially removing individual components and testing the remaining combinations on the well-defended Llama3 model using the AdvBench dataset, we quantify how each affects attack success rate and query efficiency.

5.4.2 Experimental Results and Analysis

Fig. 5 shows the ablation results, intuitively assessing each component's impact on jailbreak effectiveness. The left panel displays the Attack Success Rate and Average Query count after removing each component. Removing the content component has the most significant impact, yielding the lowest ASR and highest $\overline{Q}_{\text{verify}}$, while removing the context component has a relatively minor effect. The role and communication components show similar influence. The right panel quantifies each component's contribution using the BS metric: removing the content component reduces the BS score by approximately 0.20 (a 24.7% relative decrease); removing the context component reduces it by about 0.09 (11.1%); and removing the role or communication components results in BS scores of 0.66 and 0.65, respectively, indicating nearly equivalent impacts.

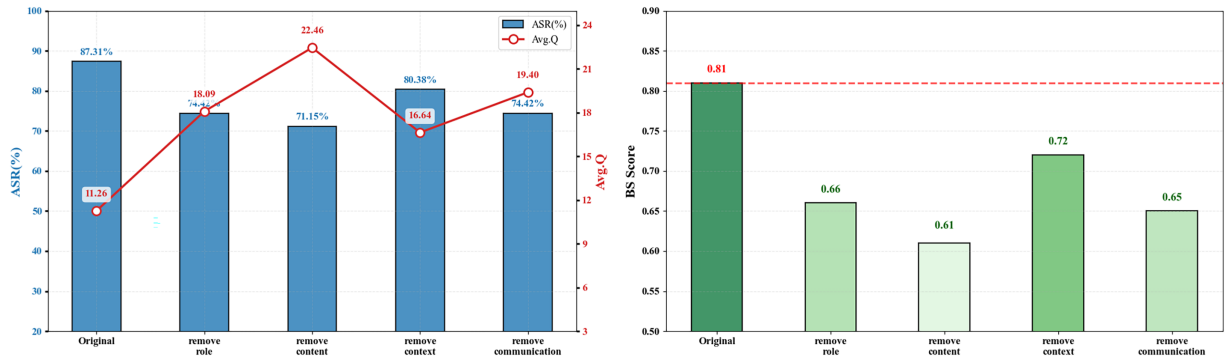

Figure 5: Ablation study. Impact of different components in the CMOOF framework on the performance of jailbreaking Llama3 model.

5.4.3 Conclusions

The study validates the modular design of CMOOF, showing each component contributes distinctly to attack performance. The Content component is most critical for effectiveness and stealth, while Context

provides limited standalone utility. The Role and Communication components have comparable, synergistic effects on persuasiveness. This confirms that attack success relies more on structured, component-driven prompt construction than on the backend model's generative capability, offering clear guidance for future jailbreak strategy optimization.

5.5 RQ5: Robustness and Generalization Analysis

5.5.1 Motivation and Definition

This section evaluates CMOOF's robustness and dual transferability to quantify the reliability of optimized solution sets for automated red teaming. Here, *generalization* refers to the ability of a single optimized strategy template to maintain effectiveness across contexts, specifically concerning: (1) **Cross-instruction generalization:** transfer to unseen malicious queries; (2) **Cross-model generalization:** transfer across diverse LLMs (e.g., Llama-3, Qwen2.5). Unlike instance-specific prompts, CMOOF optimizes reusable templates, making this dual capability the central evaluation goal.

5.5.2 Experimental Results and Analysis

We evaluate the two dimensions defined above, supplemented by a stability test.

Cross-instruction generalization. Testing five Pareto-front strategies on ten unseen instructions yielded an average semantic alignment of 3.2 (±0.4), closely matching the training score of 3.5 (±0.3). This confirms that effectiveness derives from generalizable component patterns rather than query overfitting.

Cross-model generalization. Deploying the same strategies across Llama-3-8B, Qwen2.5-7B, and Vicuna-7B resulted in a consistent ASR above 85% (Table 3). This demonstrates that component-based templates capture universal persuasion structures robust to architectural variations.

Supplementary Stability Test. Three independent runs with different seeds (Table 5) showed ASR deviations below 3% and overlapping Pareto fronts, confirming that these strategies are reproducibly discovered rather than stochastic artifacts.

Table 5: Reproducibility and stability.

Run	ASR	$\overline{Q}_{\text{verify}}$
Run 1	84.7%	10.5
Run 2	87.3%	11.1
Run 3	85.9%	12.3

5.5.3 Conclusion

CMOOF exhibits strong robustness and dual generalization, maintaining high effectiveness across both new instructions and diverse models. This supports its practicality for large-scale red teaming via reusable templates. Future work will address remaining challenges in noise elimination and broader scenario testing.

6 Threats to Validity

6.1 Internal Threats

Internal threats primarily concern factors in the research design and implementation that may affect the validity of the results. The main internal threats in our study include:

6.1.1 Experimental Design and Parameter Settings

The componentized strategy space is constructed following established methodology. Parameters are set according to the CL-GSO baseline: a population size of 15 and a maximum of 5 evolutionary generations. An adaptive mutation rate, starting at 0.7, balances exploration and exploitation. This configuration, aligned with the per-attack query budget Q_{max}, enables NSGA-II to efficiently explore the search space and converge rapidly to the Pareto front. Random initialization and repeated runs are used to control for randomness.

6.1.2 Evaluation Mechanism

This study proposes a four-level evaluation framework based on semantic alignment, using the qwen-max model to automatically assess jailbreak responses. This automated approach reduces the subjective bias of manual evaluation. By analyzing semantic consistency between queries and responses, it improves the accuracy of jailbreak success determination.

Beyond the original single-judge protocol, we have explicitly addressed two specific risks of LLM-based evaluation—*circularity* (when the judge shares failure modes with the victim) and *systematic overestimation* (when the judge is more lenient than a human annotator on borderline cases)—through three complementary mechanisms.

(i) Heterogeneous multi-judge ensemble. We augment the original qwen-max judge with two heterogeneous counterparts—GPT-4o (closed-source, OpenAI family) and Llama-3-70B-Instruct (open-source, Meta family)—using a shared four-level rubric. Success requires majority voting (at least two judges assigning Level 3), with unanimous-3-of-3 ASR reported as a conservative bound. On a 1000-sample test set, inter-judge Fleiss' κ is 0.78 (Vicuna-7B) and 0.74 (Llama-3-8B), indicating substantial agreement. All baselines are re-evaluated under this identical protocol to ensure unbiased relative rankings.

(ii) Human validation on a stratified subset. To calibrate the LLM judges against human judgment, we drew a stratified random sample of 300 attack attempts (50 per victim model × method combination, balanced across judge-agreement strata: all-agree-success, all-agree-fail, and split-decision). Three human annotators achieve substantial agreement (Cohen's $\kappa = 0.82$). The majority-judge protocol aligns with humans at ≈91% accuracy (≈6% false positive), outperforming the original single-judge protocol (≈84% accuracy, ≈11% false positive). Quantitative results show that the single-judge protocol overestimates the Attack Success Rate (ASR) by 3–4 percentage points, whereas the majority-vote protocol reduces this bias to within 1–2 percentage points. For completeness, the paper reports both the raw ASR and the human-calibrated ASR.

(iii) Cross-family circularity check. To mitigate the circularity risk, no victim model served as a judge. In addition, we performed a stricter cross-family robustness check in which all judges came from different families than the victim—e.g., when Llama-3-8B is the victim, only Qwen-max and GPT-4o serve as judges. CMOOF's ranking against the baselines is preserved under this stricter protocol, indicating that the reported efficiency advantage is not driven by judge–victim family overlap.

6.1.3 Baseline Comparison and Metric Evaluation

We select representative state-of-the-art methods (AutoDAN, CL-GSO) as baselines to ensure a fair comparison. All experiments are conducted under identical hardware and software environments, with a unified API invocation protocol to control query costs. Performance is evaluated using three core metrics: the Attack Success Rate measures the attack's effectiveness; the Average Query Count quantifies efficiency, reflecting the average resource consumption per successful jailbreak; and the Balance Score (BS)

comprehensively assesses the trade-off capability between success rate and query efficiency. Together, these metrics form a multidimensional evaluation system for comprehensive performance validation.

6.2 External Threats

External threats primarily concern the generalizability and applicability of the research conclusions in real-world scenarios. The main external threats faced by this study include:

6.2.1 Dataset Selection

This study employs AdvBench and HarmBench as evaluation benchmarks for their standardization and complementary focus: AdvBench covers traditional harmful instructions, while HarmBench targets semantically deceptive content, ensuring broad test coverage. However, existing datasets mainly cover explicit and directly harmful instructions (e.g., violence, criminal guidance) and are insufficient for more covert adversarial prompts that rely on cultural context, subtle semantics, or ideological manipulation. This limits performance evaluation and generalization of CMOOF framework in such complex threat scenarios.

6.2.2 Model Selection

Regarding model selection, this study employs open-source models (e.g., Llama-3-8B, Qwen-2.5-7B, Vicuna-7B) as target models, which facilitate analysis and avoid API-related limitations. However, it should be noted that the safety alignment of open-source models may be weaker than that of commercial models (e.g., GPT-5), which could affect the generalizability and comprehensiveness of the conclusions. Therefore, the findings of this study are primarily validated in an open-source environment, providing a foundation for future extensions to more complex models.

7 Conclusion and Future Work

This paper presents CMOOF, a framework using NSGA-II to co-optimize attack efficacy and efficiency in a structured component space. Validated via jailbreak attacks, CMOOF outperforms baselines in Success Rate and Balance Score, showing strong robustness against advanced defenses. Evaluation distinguishes between offline search cost (Q_{search}) and online verification cost ($\overline{Q}_{\text{verify}}$), establishing Pareto dominance under a unified query budget and controlled initialization. Extended-budget experiments confirm that no baseline approaches CMOOF's performance ceiling, even with eightfold more search resources. Reported success rates are calibrated via a heterogeneous three-judge ensemble and a 300-sample human-validated subset, mitigating risks of judge bias and single-model overestimation. Furthermore, we propose a layered defense pipeline—comprising component-level detection, adaptive gating, and Pareto-front safety fine-tuning—re-purposing CMOOF as a defensive synthesis tool. Building on these findings, future work will proceed along the following three dimensions:

1. **Exploring Multi-Turn Adaptive Conversational Attacks.** CMOOF's componentized strategies and multi-objective optimization align with the closed-loop process of multi-turn dialogues, making it suitable for optimizing the efficacy and cost of entire conversational sequences. It will be explored as a general conversational red teaming framework. Concurrently, the framework can be extended to more complex scenarios like multimodal and cross-lingual attacks, with its practical threat and robustness validated on commercial models like GPT-5.
2. **Investigating Dynamic Adversarial Interactions.** Current tests are limited to static defenses, risking overfitting. Future work will construct dynamic adversarial training frameworks to simulate the attack-defense process, introducing adaptive or ensemble defenses to test CMOOF's continuous adaptation

capability. This examines if it generates broadly powerful strategies rather than merely adapting to specific defenses. Concurrently, we will explore novel defenses based on semantic obfuscation to improve the security evaluation system.

3. **Developing More Fine-Grained and Standardized Evaluation Systems.** While the current four-level semantic alignment framework offers strong semantic discrimination, it faces challenges in fine-grained intent recognition and long-context assessment. Future work will add granular dimensions—semantic coherence, stealth, and graded harm—targeting culturally sensitive and ideologically manipulative threats. We will also explore converting discrete ordinal scoring into a continuous probability-based mechanism to boost optimization sensitivity and Pareto front resolution. Finally, we will build standardized benchmarks encompassing these threats to support comprehensive attack-defense research.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Authors contributed as follows: Conceptualization, Methodology, Investigation, Writing—original draft (Jialing Tao); Supervision (Song Huang, Changyou Zheng). All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The code implementing the methods described in this study is openly available in the GitHub repository: https://github.com/taoJL-2024/CMOOF. Additional data supporting the findings are available from the corresponding authors upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. OpenAI. GPT-4 technical report. arXiv:2303.08774. 2023.
2. Qiu Z, Liu W, Feng H, Liu Z, Xiao TZ, Collins KM, et al. Can large language models understand symbolic graphics programs? arXiv:2408.08313. 2025.
3. Zhang R, Han J, Liu C, Zhou A, Lu P, Qiao Y, et al. LLaMA-Adapter: efficient fine-tuning of large language models with zero-initialized attention. In: Proceedings of the International Conference on Learning Representations (ICLR 2024); 2024 May 7–11; Vienna, Austria. p. 44389–18.
4. Liu F, Zhou H, Gu B, Zou X, Huang J, Wu J, et al. Application of large language models in medicine. Nat Rev Bioeng. 2025;3(6):445–64. doi:10.1038/s44222-025-00279-5.
5. Gan W, Qi Z, Wu J, Lin JC-W. Large language models in education: vision and opportunities. In: Proceedings of the 2023 IEEE International Conference on Big Data; 2023 Dec 15–18; Sorrento, Italy. p. 4776–85.
6. Qu Y, Huang S, Yao Y. A survey on robustness attacks for deep code models. Autom Softw Eng. 2024;31(2):65. doi:10.1007/s10515-024-00464-7.
7. Huang J, Chang KC-C. Towards reasoning in large language models: a survey. In: Findings of the Association for Computational Linguistics: ACL. Kerrville, TX, USA: ACL; 2023. p. 1049–65.
8. Caffagni D, Cocchi F, Barsellotti L, Moratelli N, Sarto S, Baraldi L, et al. The revolution of multimodal large language models: a survey. In: Findings of the Association for Computational Linguistics: ACL. Kerrville, TX, USA: ACL; 2024. p. 13590–618.
9. Qu Y, Huang S, Li L, Nie P, Yao Y. Beyond intentions: a critical survey of misalignment in LLMs. Comput Mater Contin. 2025;85(1):249–300. doi:10.32604/cmc.2025.067750.
10. Qu Y, Huang S, Nie P. A review of backdoor attacks and defenses in code large language models: implications for security measures. Inf Softw Tech. 2025;182(4):107707. doi:10.1016/j.infsof.2025.107707.

11. Chao P, Robey A, Dobriban E, Hassani H, Pappas GJ, Wong E. Jailbreaking black box large language models in twenty queries. In: Proceedings of the 2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML). Piscataway, NJ, USA: IEEE; 2025. p. 23–42.
12. Deng G, Liu Y, Li Y, Wang K, Zhang Y, Li Z, et al. MasterKey: automated jailbreaking of large language model chatbots. In: Proceedings of the 2024 Network and Distributed System Security Symposium (NDSS 2024); 2024 Feb 26–Mar 1; San Diego, CA, USA. Reston, VA, USA: Internet Society; 2024
13. Zhang Z, Sun Z, Zhang Z, Guo J, He X. FC-Attack: jailbreaking multimodal large language models via auto-generated flowcharts. In: Findings of the Association for Computational Linguistics: EMNLP 2025. Kerrville, TX, USA: ACL; 2025. p. 9299–316.
14. Zou A, Wang Z, Carlini N, Nasr M, Kolter ZJ, Fredrikson M. Universal and transferable adversarial attacks on aligned language models. arXiv:2307.15043. 2023.
15. Wang X, Huang VS, Chen R, Wang H, Pan C, Sha L, et al. BlackDAN: a black-box multi-objective approach for effective and contextual jailbreaking of large language models. arXiv:2410.09804. 2024.
16. Shen X, Chen Z, Backes M, Shen Y, Zhang Y. "Do anything now": characterizing and evaluating in-the-wild jailbreak prompts on large language models. In: Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: ACM; 2024. p. 1671–85.
17. Wang Z, Xie W, Ma S, Wang E, Wang B. Evading LLMs' safety boundary with adaptive role-play jailbreaking. Electronics. 2025;14(24):4808. doi:10.3390/electronics14244808.
18. Russinovich M, Salem A, Eldan R. Great, now write an article about that: the crescendo multi-turn LLM jailbreak attack. In: Proceedings of the 34th USENIX Security Symposium; 2025 Aug 13–15; Seattle, WA, USA.
19. Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, et al. Chain-of-thought prompting elicits reasoning in large language models. In: Proceedings of the 36th Conference on Neural Information Processing Systems (NeurIPS 2022); 2022 Nov 28–Dec 9; New Orleans, LA, USA. p. 24824–37.
20. Zeng Y, Lin H, Zhang J, Yang D, Jia R, Shi W. How Johnny can persuade LLMs to jailbreak them: rethinking persuasion to challenge AI safety by humanizing LLMs. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics; 2024 Aug 11–16; Bangkok, Thailand. p. 14322–50.
21. Huang Y, Sun Y, Ruan S, Zhang Y, Dong Y, Wei X. Breaking the ceiling: exploring the potential of jailbreak attacks through expanding strategy space. In: Findings of the Association for Computational Linguistics: ACL 2025; 2025 Jul 27–Aug 1; Vienna, Austria. p. 7870–88.
22. Xu Z, Liu Y, Deng G, Li Y, Picek S. A comprehensive study of jailbreak attack versus defense for large language models. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics; 2024 Aug 11–16; Bangkok, Thailand. p. 7432–49.
23. Jin H, Chen R, Zhou A, Zhang Y, Wang H. GUARD: role-playing to generate natural-language jailbreakings to test guideline adherence of large language models. In: Proceedings of the ICLR 2024 Workshop on Secure and Trustworthy Large Language Models; 2024 May 7–11; Vienna, Austria.
24. Li X, Zhou Z, Zhu J, Yao J, Liu T, Han B. DeepInception: hypnotize large language model to be jailbreaker. arXiv:2311.03191. 2024.
25. Mehrotra A, Zampetakis M, Kassianik P, Nelson B, Anderson H, Singer Y, et al. Tree of attacks: jailbreaking black-box LLMs automatically. In: Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS 2024); 2024 Dec 9–15; Vancouver, BC, Canada.
26. Yu J, Lin X, Yu Z, Xing X. LLM-Fuzzer: scaling assessment of large language model jailbreaks. In: Proceedings of the 33rd USENIX Security Symposium; 2024 Aug 14–16; Philadelphia, PA, USA. p. 4657–74.
27. Jha P, Arora A, Ganesh V. LLM stinger: jailbreaking LLMs using RL fine-tuned LLMs. In: Proceedings of the 39th AAAI Conference on Artificial Intelligence; 2025 Feb 25–Mar 4; Vancouver, BC, Canada. p. 29393–5.
28. Liu X, Xu N, Chen M, Xiao C. AutoDAN: generating stealthy jailbreak prompts on aligned large language models. In: Proceedings of the International Conference on Learning Representations; 2024 May 7–11; Vienna, Austria. p. 56174–94.

29. Jiang F, Xu Z, Niu L, Xiang Z, Ramasubramanian B, Li B, et al. ASCII Art-based jailbreak attacks against aligned LLMs. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics; 2024 Aug 11–16; Bangkok, Thailand. p. 15157–73.
30. Liu T, Zhang Y, Zhao Z, Dong Y, Meng G, Chen K. Making them ask and answer: jailbreaking large language models in few queries via disguise and reconstruction. In: SEC '24: Proceedings of the 33rd USENIX Conference on Security Symposium; 2024 Aug 14–16; Philadelphia, PA, USA. p. 4711–28.
31. Chao P, Debenedetti E, Robey A, Andriushchenko M, Croce F, Sehwag V, et al. JailbreakBench: an open robustness benchmark for jailbreaking large language models. In: Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS 2024); 2024 Dec 10–15; Vancouver, BC, Canada. p. 55005–29.
32. Souly A, Lu Q, Bowen D, Trinh T, Hsieh E, Pandey S, et al. A StrongREJECT for empty jailbreaks. In: Proceedings of the 38th Conference on Neural Information Processing Systems (NeurIPS); 2024 Dec 10–15; Vancouver, BC, Canada. p. 125416–40.
33. Mazeika M, Phan L, Yin X, Zou A, Wang Z, Mu N, et al. HarmBench: a standardized evaluation framework for automated red teaming and robust refusal. arXiv:2402.04249. 2024.
34. Zheng Z. Implementation and evaluation of dialogue systems with BART and Qwen models. Inf Sci Eng Res. 2024;5(10):40–3. (In Chinese). doi:10.12349/iser.v5i10.3493.
35. Zheng L, Chiang W-L, Sheng Y, Zhuang S, Wu Z, Zhuang Y, et al. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In: Proceedings of the 37th Conference on Neural Information Processing Systems; 2023 Dec 10–16; New Orleans, LA, USA. p. 46595–623.
36. Robey A, Wong E, Hassani H, Pappas GJ. SmoothLLM: defending large language models against jailbreaking attacks. Trans Mach Learn Res. 2025. doi:10.48550/arxiv.2310.03684.
37. Pisano M, Ly P, Sanders A, Yao B, Wang D, Strzalkowski T, et al. Bergeron: combating adversarial attacks through a conscience-based alignment framework. arXiv:2312.00029. 2023.