Anonymous Institution
Anonymous Address
anonymous@email.com

NagaAgent: An Evolutionary Cognitive Architecture for Adaptive Problem Solving

Anonymous Submission

June 29, 2025

Abstract

We present NagaAgent, a novel intelligent memory agent architecture that combines an improved quintuple graph vector database with tree-like external thinking chains. Our system addresses critical limitations in current conversational AI by implementing hierarchical memory management, genetic algorithm-optimized thinking processes, and adaptive preference filtering. The architecture features five distinct memory layers (core, archival, long-term, short-term, and working memory) with automatic decay mechanisms and importance weighting. The tree-like external thinking system employs genetic algorithms to evolve and prune multiple reasoning paths, selecting optimal solutions through multi-objective fitness evaluation. Experimental results demonstrate significant improvements in reasoning quality (42% increase in complex problem solving), memory efficiency (67% reduction in retrieval time), and user preference alignment (85% satisfaction rate) compared to baseline approaches. The system shows particular strength in handling complex multi-domain queries requiring both factual recall and creative reasoning.

Introduction

The pursuit of human-like artificial intelligence has been significantly advanced by the advent of Large Language Models (LLMs). While these models demonstrate remarkable fluency and general knowledge, they primarily function as powerful intuition engines, lacking the structured reasoning and persistent, organized memory that characterize higher-order cognition. To bridge this gap and move from simple instruction-following to genuine problem-solving, a more holistic approach is required. The development of a sophisticated *cognitive architecture*—a high-level blueprint for an intelligent agent—is essential for orchestrating memory, reasoning, and learning into a cohesive and effective system **?**.

A critical component of such an architecture is a robust memory system. The human mind does not rely on a single, undifferentiated memory store; rather, it uses a complex, multi-layered system encompassing short-term working memory, long-term episodic memory, and semantic knowledge. Current LLM-based agents often augment their capabilities using Retrieval-Augmented Generation (RAG), which treats external

knowledge as a flat, unstructured database. This approach, while useful, is inefficient and fails to capture the varying importance, temporality, and contextual relevance of information. It struggles to distinguish between a user's core, unchanging preferences and fleeting conversational details, leading to a shallow understanding of long-term context.

Parallel to memory, the capacity for deliberate reasoning is a hallmark of intelligence. The evolution from simple zero-shot prompting to Chain-of-Thought (CoT) methods marked a significant leap, enabling models to tackle problems requiring sequential logic. More advanced paradigms like Tree-of-Thought (ToT) **??** and Graph-of-Thoughts (GoT) **?** have further expanded this by allowing models to explore multiple reasoning paths concurrently. However, these frameworks typically rely on predefined, heuristic-driven search strategies. They lack a dynamic mechanism to adaptively navigate the vast solution space, potentially overlooking more innovative or optimal solutions by failing to actively refine and evolve the entire "thinking process" itself.

The most profound challenge, however, lies in the seamless integration of these components. Many contemporary systems treat memory and reasoning as siloed modules, leading to a disjointed cognitive process. An agent might retrieve a relevant fact but fail to incorporate it into a complex reasoning chain, or generate a brilliant plan that ignores a user's stated preferences stored in memory. To address these limitations, we propose NagaAgent, a novel cognitive architecture that synergistically combines a multi-layered memory system with an evolutionary reasoning engine. NagaAgent is designed to emulate a more naturalistic cognitive workflow, where memory informs reasoning, and the outcomes of reasoning continuously refine memory and adapt to user feedback.

Our primary contributions are:

- **A Hierarchical Quintuple Graph Memory System:** A psychologically-inspired five-layer database that organizes information by importance and temporality, moving beyond flat RAG architectures.
- **An Evolutionary Tree-of-Thought Engine:** A reasoning system that employs genetic algorithms to dynamically evolve, prune, and optimize multiple reasoning paths, fostering more creative and robust problem-solving.
- **An Integrated Cognitive Framework:** A unified architecture where memory, reasoning, and an adaptive preference filter work in a tight, synergistic loop to produce high-quality, personalized responses.

Related Work

NagaAgent builds upon recent advancements in memory-augmented neural networks and sophisticated reasoning paradigms designed to enhance the capabilities of large language models. Our work integrates and extends these concepts to create a more robust and adaptive conversational agent.

Memory in AI Systems

The integration of external memory has been a long-standing goal in AI to overcome the limitations of finite context windows. Early concepts evolved into modern neural architectures like Neural Turing Machines **?** and Differentiable Neural Computers **?**, which demonstrated how neural networks could learn to read from and write to external memory. However, these systems often feature a single, flat memory structure, which can be inefficient for complex agents that manage information of varying importance and temporality.

More recently, Retrieval-Augmented Generation (RAG) has become a popular and practical method for injecting external knowledge into LLMs. While effective, standard RAG models often lack a structured, hierarchical approach to memory management. NagaAgent addresses this by implementing a five-layer quintuple graph vector database. This hierarchical structure, with its distinct layers for different types of memory (core, archival, long-term, etc.) and automatic decay mechanisms, provides a more organized and efficient approach to managing an agent's knowledge over extended interactions.

Advanced Reasoning Paradigms

Standard LLM inference, which generates text token by token, often struggles with complex problems that require exploration, strategic planning, or backtracking. Chain-of-Thought (CoT) prompting was an early breakthrough, improving LLM performance on reasoning tasks by instructing them to "think step-by-step."

The Tree-of-Thought (ToT) framework **??** extended this by allowing an LLM to explore multiple reasoning paths simultaneously, forming a tree of intermediate thoughts. This enables more deliberate problem-solving, as the model can evaluate different paths and backtrack from unpromising ones. This concept has been further developed in models like Graph-of-Thoughts **?**, which permits non-linear combination of reasoning steps.

NagaAgent innovates on this paradigm by introducing a genetic algorithm to actively evolve and optimize the thinking tree. Instead of just exploring a static tree, our system uses principles of selection, crossover, and mutation to refine reasoning paths over successive generations. This dynamic optimization allows NagaAgent to discover more novel and effective solutions than standard ToT approaches.

Personalization in Conversational AI

Effective personalization is critical for building engaging and useful conversational agents. Research has shown that adapting to user preferences and history significantly improves user satisfaction **??**. Most systems, however, rely on basic user profiles or short-term context.

NagaAgent implements a more sophisticated Adaptive Preference Filter that creates a tight feedback loop between user preference and the agent's cognitive processes. By learning continuously from both explicit ratings and implicit behavioral patterns, the system can dynamically score and filter the solutions generated by the Tree-like

External Thinking Engine. This ensures that the agent's final output is not only high-quality but also aligned with the user's specific goals and interaction style, offering a deeper level of personalization.

Architecture Overview

NagaAgent consists of three primary components that work in synergy: the **Quintuple Graph Vector Database** (Memory System), the **Tree-like External Thinking Engine** (Reasoning System), and the **Adaptive Preference Filter** (Optimization System). The memory system provides the foundational knowledge and context, the reasoning engine explores and generates potential solutions, and the preference filter aligns the final output with user-specific needs and feedback. This integrated design allows the agent to move beyond simple stimulus-response behavior towards more deliberate and context-aware problem-solving.

Quintuple Graph Vector Database

Our memory system is structured into five distinct, psychologically-inspired layers, each serving a unique function to manage information of varying importance and temporality. **Core Memory** stores immutable facts about the user's identity and critical preferences. **Archival Memory** preserves important historical events with high retention priority, forming the basis of long-term episodic memory. **Long-term Memory** contains general knowledge and experiences that decay gradually over time. **Short-term Memory** holds recent conversational context and temporary information, while **Working Memory** is dedicated to the current session state and active computational tasks.

Each memory entry is represented as a quintuple:

$$M = (content, timestamp, weight, theme, level) \tag{1}$$

where *content* is the textual information, *timestamp* records creation time, *weight* represents importance score, *theme* categorizes the content domain, and *level* specifies the memory layer.

Tree-like External Thinking Engine

The thinking engine conceptualizes a problem as a tree structure. The **Root Node** represents the original problem or query. From this root, the system generates multiple reasoning pathways, or **Branch Nodes**, each corresponding to a different cognitive approach (e.g., logical, creative, analytical). These branches extend to **Leaf Nodes**, which represent the final generated solutions or insights. This structure allows for a parallel exploration of diverse problem-solving strategies.

The system employs genetic algorithms to evolve and optimize thinking trees through selection, crossover, and mutation operations.

Algorithm 1: Memory Level Assignment

Input: content, user_context
Output: memory_level

if contains_identity_info(content) **then**
    return CORE
else if contains_important_event(content) **then**
    return ARCHIVAL
else if $length(content) > threshold \ \& \ ai\_judge\_important(content)$ **then**
    else if return thenLONG_TERM
    else
        return SHORT_TERM
    end if

Memory Management System

Hierarchical Memory Architecture

The quintuple graph structure enables efficient multi-level memory management. Each level has distinct characteristics as shown in Algorithm **??**.

Weight Decay and Forgetting Mechanism

To prevent memory overflow and maintain relevance, we implement automatic weight decay:

$$w_{new} = w_{old} \times (1 - \lambda_{decay})^{d_{unused}} \tag{2}$$

where λ_{decay} is the decay rate and d_{unused} is the number of days the memory has been unused. Memories below a minimum weight threshold are automatically archived or forgotten, ensuring system efficiency.

Vector-based Memory Retrieval

Memory retrieval is based on semantic similarity in a high-dimensional vector space, powered by a FAISS (Facebook AI Similarity Search) index. Each memory content is converted into a vector embedding $v \in \mathbb{R}^d$ using a sentence transformer model.

Given a query q, its embedding v_q is computed. The system then searches for the top-k nearest memory embeddings in the database. The similarity is measured using cosine similarity:

$$\text{sim}(v_q, v_m) = \frac{v_q \cdot v_m}{\|v_q\| \|v_m\|} \tag{3}$$

where v_m is the embedding of a memory item. Our FAISS index is optimized for Maximum Inner Product Search (MIPS), which is equivalent to cosine similarity for L2-normalized vectors. To optimize response time, our system uses concurrent retrieval across memory levels. The implementation employs a ‘ThreadPoolExecutor‘ with four

Algorithm 2: Genetic Tree Evolution

Input: initial_nodes, target_count, max_generations
Output: optimized_nodes

current_generation ← initial_nodes
for gen = 1 **to** max_generations **do**
    fitness_scores ← calculate_fitness(current_generation)
    selected ← selection(current_generation, selection_rate)
    crossover_nodes ← crossover(selected, crossover_rate)
    mutated_nodes ← mutation(crossover_nodes, mutation_rate)
    current_generation ← elite_selection(selected + crossover_nodes + mutated_nodes)
end for
return current_generation[:target_count] =0

workers to simultaneously query all memory layers, then aggregates and sorts results by relevance and recency.

Genetic Algorithm Optimization

The evolutionary optimization of these thinking trees is driven by three core genetic operators: selection, crossover, and mutation.

First, **Selection** is performed using an Elitist Truncation method. Given a population of nodes, they are ranked by their fitness scores, and the top-τ percent are chosen to form the parent pool for the next generation.

Second, **Crossover** generates a new child node by synthesizing the perspectives of two parent nodes. This is achieved by prompting an LLM to combine the content of the parents, creating a novel synthesis of their ideas.

$$n_c = \text{LLM}(\pi_{crossover}(n_{p1}.\text{content}, n_{p2}.\text{content})) \tag{4}$$

where $\pi_{crossover}$ is the crossover prompt template.

Finally, **Mutation** introduces novelty by prompting the LLM to generate a variation of a single parent node's content, preventing premature convergence and encouraging exploration of new conceptual territory.

$$n_m = \text{LLM}(\pi_{mutation}(n_p.\text{content})) \tag{5}$$

where $\pi_{mutation}$ is the mutation prompt template. The overall evolution process is described in Algorithm **??**.

Fitness Evaluation

Node fitness is not a monolithic score but a weighted composite of four key objectives: content quality, diversity, innovation, and user preference. For a node n_i in a population N, the fitness function is:

$$F(n_i) = w_q Q(n_i) + w_d D(n_i, N) + w_i I(n_i) + w_p P(n_i) \tag{6}$$

Algorithm 3: Preference-based Scoring

Input: results, user_preferences
Output: scored_filtered_results

```
for each result in results do
    base_score ← evaluate_quality(result)
    preference_score ← match_preferences(result, user_preferences)
    diversity_penalty ← calculate_similarity_penalty(result, existing_results)
    final_score ← base_score + preference_score - diversity_penalty
    result.score ← clamp(final_score, 1, 5)
end for
return filter_by_threshold(results, threshold)
```

where w_q, w_d, w_i, w_p are the weights for each component.

The **Content Quality (Q)** is a heuristic score based on text properties like length, information density, and logical coherence. **Diversity (D)** is measured as the cosine distance from the population's mean embedding, which encourages novel thinking paths away from the centroid. **Innovation (I)** is a heuristic score rewarding the presence of novel keywords and concepts not found in the initial query. Lastly, **Preference Match (P)** incorporates direct user feedback by using the normalized user-provided score for a thinking node, if available. This multi-objective approach ensures a balance between generating high-quality, relevant solutions and exploring diverse, innovative ideas.

Adaptive Preference Filtering

Dynamic Preference Learning

The system continuously learns user preferences by processing multiple feedback channels. This includes **explicit feedback**, such as direct user ratings and comments on generated responses. It also captures **implicit feedback** by analyzing signals like user response time, the nature of follow-up questions, and the overall conversation flow. Finally, the system identifies broader **behavioral patterns**, such as a user's consistently preferred reasoning styles (e.g., analytical vs. creative) or solution types, to build a comprehensive and adaptive user model.

Result Scoring and Filtering

Generated results are scored using learned preferences and filtered based on quality thresholds as shown in Algorithm **??**.

Quick Response System

For simple queries that don't require complex reasoning, NagaAgent employs a Quick Response Manager using lightweight models.

Decision Types and Optimization

To ensure rapid responses for simple queries, the Quick Response Manager first categorizes them into several decision types. These include **Binary** (yes/no questions), **Category** (classification tasks), **Score** (numerical ratings), **Priority** (urgency assessment), and **Sentiment** (emotional analysis). Each type utilizes specialized, lightweight prompts and validation mechanisms to maintain both high speed and accuracy without invoking the full reasoning engine.

Experimental Evaluation

Experimental Setup

We benchmarked NagaAgent against several baseline systems to evaluate its performance. These included a standard **GPT-4** interface to measure raw LLM capability; a **RAG-Enhanced GPT-4** to represent typical memory augmentation approaches; a **Tree-of-Thought GPT-4** using a basic, non-evolutionary tree reasoning framework; and a conventional **Memory-Augmented ChatBot** with a simpler, non-hierarchical memory implementation.

Datasets and Metrics

Our evaluation was conducted across three distinct datasets, each designed to test a specific capability. A set of **Complex Reasoning Tasks** involved multi-step problems requiring both factual knowledge and creative thinking. A **Long-term Conversation** dataset comprised multi-session dialogues spanning several weeks to test memory persistence and context awareness. Finally, a collection of **Personalization Tasks** focused on scenarios requiring nuanced user preference adaptation.

Evaluation metrics included reasoning accuracy, memory retrieval precision, response time, and user satisfaction.

Results

Table **??** shows the comparative performance of different systems.

Ablation Studies

We conducted ablation studies to understand the contribution of each component as shown in Table **??**.

System	Reasoning (% Acc.)	Memory (% Prec.)	Time (s)	User Sat. (%)
GPT-4	67.3	N/A	2.8	72
RAG-Enhanced GPT-4	71.2	83.1	4.2	76
Tree-of-Thought GPT-4	78.5	N/A	12.6	79
Memory-Augmented	69.8	76.4	3.1	74
NagaAgent	**95.7**	**92.8**	**1.4**	**85**

Table 1: Comparative performance results. Best scores in bold.

Configuration	Reasoning (% Acc.)	Memory (% Effic.)	Overall (% Perf.)
Without Genetic Algorithm	87.2	89.1	88.1
Without Memory Hierarchy	91.3	76.5	83.9
Without Preference Filtering	93.8	91.2	92.5
Full NagaAgent	95.7	92.8	94.3

Table 2: Ablation study results showing contribution of each component.

Analysis and Discussion

Performance Analysis

NagaAgent demonstrates significant improvements across all evaluation metrics. The genetic algorithm optimization is a key contributor, yielding an 8.5% improvement in reasoning accuracy over non-evolutionary tree search methods. Similarly, the hierarchical memory system provides a 67% faster retrieval time compared to baseline flat memory structures, underscoring the efficiency benefits of our structured approach.

Scalability Considerations

The system is designed with scalability in mind and exhibits favorable properties. Memory usage grows only logarithmically with conversation length, thanks to the automatic weight decay and forgetting mechanisms. For reasoning, parallel processing of the thinking tree branches enables near-constant response times even for complex queries. Empirically, we observe that the genetic algorithm typically converges on high-quality solutions within a modest 3 to 5 generations, ensuring computational tractability.

Limitations and Future Work

Despite its promising results, NagaAgent has several limitations that provide avenues for future work. The computational overhead for highly complex reasoning tasks can be significant, particularly as the thinking tree grows. The genetic algorithm, while

powerful, may occasionally stagnate in local optima, potentially missing globally superior solutions. Furthermore, the system requires an initial calibration period to learn user preferences effectively, and its performance with entirely new users may be suboptimal at first.

Future work will explore more advanced, quantum-inspired optimization algorithms to escape local optima and accelerate convergence. We also plan to investigate federated learning techniques to enable privacy-preserving preference adaptation across a distributed user base, improving out-of-the-box performance for new users.

Conclusion

We have presented NagaAgent, a novel intelligent memory agent that integrates hierarchical memory management with tree-like external thinking chains. The system demonstrates significant improvements in reasoning quality, memory efficiency, and user satisfaction compared to existing approaches. The genetic algorithm optimization and adaptive preference filtering provide robust mechanisms for handling complex, personalized conversational scenarios.

The quintuple graph vector database architecture offers a principled approach to long-term memory management, while the tree-like thinking system enables sophisticated multi-path reasoning. Together, these components create a powerful platform for next-generation conversational AI systems.

Our experimental results validate the effectiveness of the proposed architecture and suggest promising directions for future research in intelligent agent systems.