The Iterative Thought Refinement System: A Novel Architecture for Emergent AI Reasoning through Dynamic Large Language Model-Driven Decision Making and Knowledge Graph Integration

Thomas "Thom" Heinrich
Independent Research
th@thomheinrich.com

June 13, 2025

Abstract

We present the Iterative Thought Refinement System (ITRS), a groundbreaking architecture that revolutionizes artificial intelligence reasoning through a purely large language model (LLM)-driven iterative refinement process integrated with dynamic knowledge graphs and semantic vector embeddings. Unlike traditional heuristic-based approaches, ITRS employs zero-heuristic decision making, where all strategic choices emerge from LLM intelligence rather than hardcoded rules. The system introduces six distinct refinement strategies (TARGETED, EXPLORATORY, SYNTHESIS, VALIDATION, CREATIVE, and CRITICAL), a persistent thought document structure with semantic versioning, and real-time thinking step visualization. Through synergistic integration of knowledge graphs for relationship tracking, semantic vector engines for contradiction detection, and dynamic parameter optimization, ITRS achieves convergence to optimal reasoning solutions while maintaining complete transparency and auditability. We demonstrate the system's theoretical foundations, architectural components, and potential applications across explainable AI (XAI), trustworthy AI (TAI), and general LLM enhancement domains. The theoretical analysis demonstrates significant potential for improvements in reasoning quality, transparency, and reliability compared to single-pass approaches, while providing formal convergence guarantees and computational complexity bounds. The architecture advances the state-of-the-art by eliminating the brittleness of rule-based systems and enabling truly adaptive, context-aware reasoning that scales with problem complexity.

Keywords: Iterative reasoning, large language models, knowledge graphs, semantic embeddings, explainable AI, trustworthy AI, emergent reasoning, zero-heuristic systems

1 Introduction

The rapid advancement of large language models (LLMs) has fundamentally transformed the landscape of artificial intelligence, yet significant challenges remain in achieving robust, transparent, and reliable reasoning systems. Current state-of-the-art models, while demonstrating remarkable capabilities across diverse domains, often struggle with maintaining coherent reasoning chains over extended problem-solving sessions, detecting and resolving self-contradictions, and providing transparent, auditable explanations for their conclusions [Wang et al., 2022, Wei et al., 2022]. These limitations become particularly pronounced when addressing complex, multi-faceted problems that require sustained analytical thinking, iterative refinement, and the integration of diverse knowledge sources.

Traditional approaches to enhancing LLM reasoning have largely relied on structured prompting techniques such as Chain-of-Thought (CoT) prompting [Wei et al., 2022], Tree-of-Thoughts [Yao et al., 2023], and Graph-of-Thoughts [Besta et al., 2024a]. While these methods have demonstrated substantial improvements over single-pass inference, they remain fundamentally limited by their dependence on predefined heuristics, fixed exploration strategies, and hardcoded decision rules. Such approaches lack the adaptability and contextual sensitivity required for optimal performance across the diverse spectrum of reasoning tasks encountered in real-world applications.

In this paper, we introduce the Iterative Thought Refinement System (ITRS), a novel architecture that fundamentally reimagines how AI systems approach complex reasoning tasks. ITRS treats reasoning not as a linear process but as an iterative art form, continuously refining a persistent thought document through strategic iterations until convergence criteria are met or computational budgets are exhausted. The system's revolutionary approach lies in its complete elimination of hardcoded heuristics—every decision, from strategy selection and parameter optimization to convergence assessment and termination criteria, emerges purely from LLM intelligence.

1.1 Core Innovations and Contributions

The fundamental innovation of ITRS centers on three synergistic architectural components that work in concert to enable emergent reasoning capabilities:

1. **Zero-Heuristic Decision Architecture**: ITRS eliminates all hardcoded decision rules and parameters in favor of a pure LLM-driven approach. Unlike traditional systems that employ fixed strategies such as "if confidence < 0.5 then use exploratory mode," ITRS delegates all strategic decisions to the LLM itself, enabling adaptive reasoning that responds to subtle contextual cues impossible to capture in predetermined rules.

2. **Persistent Thought Document with Semantic Versioning**: The system maintains a structured thought document serving as persistent reasoning memory with nine distinct sections tracking different aspects of understanding. Each section maintains detailed confidence scores and semantic versioning to track evolution across iterations, providing complete transparency into the reasoning process.

3. **Synergistic Knowledge Representation**: ITRS integrates dynamic knowledge graphs for capturing logical relationships between thoughts with state-of-the-art semantic vector embeddings for detecting contradictions, identifying gaps, and suggesting exploration directions. This dual representation enables both symbolic and subsymbolic reasoning processes.

1.2 Research Contributions

This work makes several significant contributions to the field of AI reasoning systems:

- We present the first fully LLM-driven iterative reasoning architecture that eliminates hardcoded heuristics while maintaining formal convergence guarantees.
- We introduce a novel approach to knowledge graph construction that captures the evolution of reasoning rather than static factual knowledge, enabling real-time contradiction detection and insight synthesis.
- We demonstrate how semantic embeddings can be integrated into the reasoning loop not just for retrieval but for active guidance of the thinking process.
- We provide formal theoretical analysis of convergence properties and computational complexity bounds for the ITRS architecture.
- We present comprehensive applications across explainable AI, trustworthy AI, and general LLM enhancement, demonstrating the versatility and effectiveness of the approach.

The remainder of this paper is organized as follows: Section 2 reviews related work in iterative reasoning, knowledge graph integration, and semantic embeddings. Section 3 presents the complete ITRS architecture with detailed component descriptions. Section 4 provides theoretical analysis including convergence properties and complexity bounds. Section 5 demonstrates applications across multiple use cases. Section 6 outlines a framework for future empirical validation. Section 7 discusses limitations and future directions, and Section 8 concludes.

2 Related Work

2.1 Evolution of Iterative Reasoning in Large Language Models

The field of LLM reasoning has witnessed rapid evolution from simple prompt engineering to sophisticated iterative approaches. The seminal work by Wei et al. [2022] introduced Chain-of-Thought prompting, demonstrating that encouraging models to produce explicit reasoning steps significantly improves performance on complex reasoning tasks. This breakthrough was further enhanced by Wang et al. [2022], who introduced self-consistency methods that sample multiple reasoning paths and select the most consistent answer.

Building upon these foundations, Yao et al. [2023] proposed Tree-of-Thoughts, which generalizes chain-of-thought reasoning to explore multiple reasoning branches in a tree structure. This approach enables more systematic exploration of the solution space by maintaining and evaluating multiple reasoning paths simultaneously. The work was further extended by Besta et al. [2024a], who introduced Graph-of-Thoughts, allowing for arbitrary graph structures in reasoning exploration.

However, these approaches share a fundamental limitation: they rely heavily on predefined heuristics for path selection, evaluation, and termination. For instance, Tree-of-Thoughts uses fixed breadth and depth parameters, while Graph-of-Thoughts employs predetermined aggregation strategies. ITRS addresses this limitation by replacing all such heuristics with emergent LLM intelligence, enabling truly adaptive reasoning that scales with problem complexity.

Recent work by Besta et al. [2024b] provides a comprehensive survey of these reasoning paradigms, highlighting the need for more flexible and adaptive approaches. Our work directly addresses this need by introducing the first fully emergent reasoning architecture.

2.2 Knowledge Graph Integration with Large Language Models

The integration of knowledge graphs with LLMs has emerged as a promising direction for grounding reasoning in structured knowledge and reducing hallucinations. Pan et al. [2023] provide a comprehensive roadmap for unifying LLMs and knowledge graphs, identifying key challenges and opportunities in this intersection.

Several recent works have explored novel approaches to knowledge graph-enhanced reasoning. Markowitz et al. [2024] introduced Tree-of-Traversals, a zero-shot reasoning algorithm that augments black-box language models with knowledge graphs through strategic traversal paths. This work demonstrates how external knowledge structures can guide reasoning without requiring model fine-tuning.

Liu et al. [2024] developed a path selection mechanism for knowledge graph-enhanced LLMs, showing how careful selection of relevant knowledge graph paths can improve reasoning accuracy. Similarly, Jin et al. [2024] introduced Graph Chain-of-Thought, which augments traditional CoT reasoning with graph-structured knowledge.

However, these approaches primarily focus on leveraging existing, static knowledge graphs. ITRS takes a fundamentally different approach by dynamically constructing knowledge graphs during the reasoning process itself, capturing not external knowledge but the evolving relationships between thoughts and insights. This enables real-time detection of contradictions, identification of convergent insights, and tracking of reasoning chains across iterations.

2.3 Semantic Embeddings in Reasoning Systems

The application of semantic embeddings to reasoning tasks has gained significant attention, particularly in the context of retrieval-augmented generation and memory systems. Gutiérrez et al. [2024] introduced HippoRAG, a neurobiologically-inspired approach to long-term memory for LLMs that uses semantic similarity for memory retrieval and organization.

More recently, Gutiérrez et al. [2025] extended this work with HippoRAG 2, demonstrating how non-parametric continual learning can be achieved through sophisticated embedding-based memory systems. He et al. [2024] developed G-Retriever, showing how retrieval-augmented generation can be enhanced with graph-structured embeddings for textual understanding and question answering.

While these works demonstrate the power of semantic embeddings for information retrieval and organization, ITRS integrates embeddings directly into the reasoning loop itself. Our semantic vector engine not only organizes thoughts but actively guides the reasoning process by identifying semantic gaps, detecting redundancies, and suggesting exploration directions based on embedding space topology.

2.4 Self-Refinement and Iterative Improvement

The concept of iterative self-improvement in AI systems has deep roots, with recent work focusing specifically on LLM self-refinement. Madaan et al. [2023] introduced Self-Refine, demonstrating how models can iteratively improve their outputs through self-generated feedback. This approach shows promise for improving solution quality without external supervision.

Bai et al. [2022] developed Constitutional AI, which uses AI feedback for harmlessness training, demonstrating how models can learn to improve their behavior through iterative refinement processes. Xiong et al. [2024] extended these ideas to agent learning, showing how step-level process refinement can improve LLM agent performance.

More recently, Han et al. [2024] introduced MERLIN, which applies iterative refinement to multimodal embedding systems for text-video retrieval tasks, demonstrating the broad applicability of iterative improvement approaches.

ITRS builds upon these foundations but introduces several key innovations: (1) elimination of hardcoded refinement strategies in favor of emergent strategy selection, (2) integration of knowledge graphs and semantic embeddings into the refinement loop, and (3) formal convergence guarantees with adaptive termination criteria.

2.5 Explainable and Trustworthy AI Frameworks

The growing deployment of AI systems in critical applications has intensified research into explainable and trustworthy AI. Parekh et al. [2024] developed concept-based explainability frameworks for large multimodal models, while Salih et al. [2023] analyzed the effectiveness of popular explainability methods like SHAP and LIME.

In the domain of trustworthy AI, Li et al. [2023] provide a comprehensive survey of principles and practices, while Liang et al. [2022] examine the data challenges in creating trustworthy AI systems. Kowald et al. [2024] present evaluation frameworks for trustworthy AI, and Li et al. [2024] explore human-AI trust relationships.

ITRS contributes to this field by providing unprecedented transparency through its iterative reasoning traces, comprehensive audit trails, and confidence mapping capabilities. The system's architecture naturally supports both explainability and trustworthiness requirements through its structured thought documentation and relationship tracking.

2.6 Formal Verification in AI Systems

The need for formal guarantees in AI systems has led to significant research in neural network verification. Katz et al. [2017] introduced Reluplex, an efficient SMT solver for verifying deep neural networks, while Albarghouthi [2021] provides a comprehensive introduction to neural network verification techniques.

Recent work by Lopez et al. [2023] developed NNV 2.0, a neural network verification tool, and Katz et al. [2019] introduced the Marabou framework for verification and analysis of deep neural networks.

While ITRS does not provide formal verification in the traditional sense, it offers convergence guarantees and computational complexity bounds that provide theoretical foundations for reasoning reliability.

3 System Architecture

3.1 Architectural Overview

The ITRS architecture represents a paradigm shift from traditional rule-based reasoning systems to a fully emergent, LLM-driven approach. The system is built around four core architectural principles:

1. **Emergent Intelligence**: All decisions emerge from LLM reasoning rather than hardcoded heuristics

2. **Persistent Memory**: Structured thought documents maintain reasoning state across iterations
3. **Dual Knowledge Representation**: Integration of symbolic (knowledge graphs) and sub-symbolic (embeddings) representations
4. **Adaptive Convergence**: Dynamic assessment of reasoning progress and termination criteria

3.2 Core Components

3.2.1 Thought Document Structure

The central data structure in ITRS is the thought document, a sophisticated representation that maintains the complete state of reasoning across iterations. The document is organized into nine distinct sections, each serving a specific cognitive function:

UNDERSTANDING Current comprehension of the problem, including problem statement interpretation, constraint identification, and scope definition.

HYPOTHESES Working theories and assumptions under consideration, including alternative explanations and speculative connections.

INSIGHTS Key discoveries, realizations, and breakthrough moments that advance understanding of the problem.

OPEN_QUESTIONS Unresolved aspects requiring further exploration, including identified knowledge gaps and areas of uncertainty.

SOLUTION_APPROACHES Methods, strategies, and algorithmic approaches under consideration for problem resolution.

EVIDENCE Supporting facts, validation results, and empirical observations that ground the reasoning process.

SYNTHESIS Integrated understanding and conclusions that emerge from combining insights across sections.

CONFIDENCE_MAP Detailed confidence assessment for each component, including uncertainty quantification and reliability estimates.

META_REFLECTION Self-assessment of reasoning quality, including identification of potential biases and methodological concerns.

Each section maintains fine-grained confidence scores in the range $[0.0, 1.0]$ and employs semantic versioning to track evolution across iterations. This structure enables comprehensive tracking of reasoning development while maintaining transparency and auditability.

3.2.2 Refinement Strategy Framework

ITRS implements six distinct refinement strategies, each designed to address specific aspects of the reasoning process:

```python
class RefinementStrategy(Enum):
    """
    Enumeration of available refinement strategies for iterative
    thought improvement.
    """
    TARGETED = "targeted"         # Precision-focused improvement
    EXPLORATORY = "exploratory" # Creative exploration and discovery
    SYNTHESIS = "synthesis"      # Integration of disparate insights
    VALIDATION = "validation"    # Critical testing and verification
    CREATIVE = "creative"         # Innovative and lateral thinking
    CRITICAL = "critical"         # Devil's advocate analysis
```

Listing 1: Refinement Strategy Enumeration

Each strategy is associated with specific cognitive patterns and is selected dynamically based on the current state of the reasoning process:

TARGETED Focuses on precision improvement and detailed refinement of specific sections or insights. This strategy is typically selected when the reasoning has identified specific areas needing enhancement.

EXPLORATORY Emphasizes creative exploration and discovery of new avenues of investigation. Selected when the reasoning process has reached a plateau or when novel approaches are needed.

SYNTHESIS Concentrates on integrating disparate insights and building coherent understanding across different aspects of the problem. Chosen when multiple insights need to be unified.

VALIDATION Applies critical testing and verification to existing insights and conclusions. Selected when the reasoning needs to ensure reliability and detect potential errors.

CREATIVE Employs innovative and lateral thinking approaches to break through conceptual barriers. Used when conventional approaches have been exhausted.

CRITICAL Implements devil's advocate analysis to challenge assumptions and test robustness of conclusions. Applied to ensure comprehensive consideration of alternatives.

3.2.3 Knowledge Graph Engine

The knowledge graph component of ITRS dynamically constructs and maintains a graph representation of relationships between thoughts, insights, and conclusions as they evolve across iterations. Unlike traditional knowledge graphs that encode static factual knowledge, the ITRS graph captures the dynamic relationships between reasoning elements.

The graph employs eleven distinct relationship types, each capturing a different aspect of reasoning connectivity:

Table 1: Knowledge Graph Relationship Types in ITRS

Relation Type	Description
SUPPORTS	Evidence or reasoning that strengthens a claim
CONTRADICTS	Direct logical opposition between statements
BUILDS_ON	Extension or elaboration of an existing idea
ANSWERS	Direct response to a posed question
RAISES	Generation of new questions or uncertainties
REFINES	Improvement in precision or accuracy
CONNECTS	Bridge or link between disparate concepts
VALIDATES	Independent confirmation or verification
CHALLENGES	Questioning of assumptions or conclusions
EXPLAINS	Causal or mechanistic explanation
REQUIRES	Logical or practical dependency

The knowledge graph engine implements sophisticated algorithms for:

- **Contradiction Detection**: Identifying logical inconsistencies across the reasoning space through graph traversal and semantic analysis.
- **Convergent Insight Discovery**: Finding insights that are supported by multiple independent reasoning paths, indicating robust conclusions.
- **Orphan Thought Identification**: Detecting isolated thoughts that lack sufficient connections to the broader reasoning framework.
- **Relationship Strength Assessment**: Quantifying the strength and reliability of connections between reasoning elements.

3.2.4 Semantic Vector Engine

The semantic vector engine provides continuous, high-dimensional representations of thoughts and insights using state-of-the-art transformer-based embeddings. The current implementation utilizes the sentence-transformers/all-mpnet-base-v2 model, which generates 768-dimensional embeddings optimized for semantic similarity tasks.

The semantic engine provides four primary analytical capabilities:

1. **Real-time Contradiction Detection**: By analyzing semantic embeddings, the system can identify potential contradictions even when they are not explicitly logical, capturing subtle inconsistencies in meaning and implication.
2. **Semantic Gap Identification**: Using HDBSCAN clustering and embedding space analysis, the system identifies regions of conceptual space that remain unexplored, suggesting directions for further investigation.
3. **Coherence Scoring**: The system computes coherence metrics across the thought space, identifying areas where reasoning may be fragmented or disconnected.
4. **Exploration Direction Suggestion**: Based on embedding topology and density analysis, the system suggests specific directions for expanding the reasoning process.

3.3 Operational Flow and Decision Making

The ITRS reasoning process follows a sophisticated iterative loop that adapts dynamically to the evolving state of the reasoning process. Each iteration consists of nine distinct phases:

Algorithm 1 ITRS Iterative Reasoning Loop

Input: Query q, maximum iterations I_{max}, convergence threshold ϵ
Initialize thought document D_0 from query q
Set iteration counter $t \leftarrow 0$
while $t < I_{max}$ and not converged **do**
  $G_t \leftarrow$ AnalyzeGaps(D_t, knowledge graph, embeddings)
  $S_t \leftarrow$ SelectStrategy(D_t, G_t, t)
  $W_t \leftarrow$ IdentifyWeakSections(D_t, G_t, S_t)
  $\theta_t \leftarrow$ OptimizeParameters(S_t, D_t, G_t)
  $D_{t+1} \leftarrow$ ExecuteRefinement(D_t, S_t, W_t, θ_t)
  UpdateKnowledgeGraph(D_{t+1}, D_t)
  UpdateSemanticEmbeddings(D_{t+1})
  $c_t \leftarrow$ AssessConvergence(D_{t+1}, D_t, ϵ)
  $t \leftarrow t + 1$
end while
Return: D_t, convergence status, iteration count

3.3.1 Gap Analysis and Strategic Planning

The gap analysis phase represents one of the most critical innovations in ITRS. Rather than relying on predetermined criteria for identifying areas needing improvement, the system employs sophisticated LLM-driven analysis that considers multiple dimensions:

- **Query Coverage Analysis**: Assessing whether the current reasoning directly addresses all aspects of the original query.
- **Breadth Assessment**: Determining if the reasoning has become too narrowly focused or if important perspectives have been overlooked.
- **Logical Gap Identification**: Finding missing logical steps or unsupported inferential leaps.
- **Section Utilization Analysis**: Identifying underutilized sections of the thought document that may contain important unexplored potential.
- **Redundancy Detection**: Recognizing repetitive or circular reasoning patterns that do not advance understanding.

3.3.2 Zero-Heuristic Strategy Selection

The strategy selection mechanism represents the core innovation of ITRS's zero-heuristic approach. Traditional systems might employ rules such as:

```python
# Traditional approach - what ITRS explicitly avoids
if confidence_score < 0.5:
    strategy = "EXPLORATORY"
    temperature = 0.8
elif logical_gaps_detected > 3:
    strategy = "VALIDATION"
    temperature = 0.3
else:
    strategy = "SYNTHESIS"
    temperature = 0.5
```

Listing 2: Traditional Heuristic Approach (Not Used in ITRS)

ITRS replaces all such heuristics with emergent LLM decision making:

```python
async def select_optimal_strategy(
    self,
    doc: ThoughtDocument,
    gaps: Dict[str, Any],
    iteration: int
) -> RefinementStrategy:
    """
    Use LLM intelligence to select the optimal refinement strategy
    based on current document state, identified gaps, and iteration
    context.

    Args:
        doc: Current thought document state
        gaps: Identified gaps from analysis phase
        iteration: Current iteration number

    Returns:
        Optimal refinement strategy for current context
    """
    strategy_prompt = f"""
    Analyze the current reasoning state and select the optimal
    refinement strategy.

    CURRENT DOCUMENT STATE:
    {doc.to_detailed_analysis()}

    IDENTIFIED GAPS:
    {gaps}

    ITERATION CONTEXT: {iteration}

    AVAILABLE STRATEGIES:
    - TARGETED: Precision-focused improvement
    - EXPLORATORY: Creative exploration
    - SYNTHESIS: Integration of insights
    - VALIDATION: Critical testing
    - CREATIVE: Innovative thinking
    - CRITICAL: Devil's advocate analysis

    Select the single most appropriate strategy and provide reasoning.
    """

    # LLM provides strategic decision with justification
    response = await self.llm.generate(strategy_prompt)
```

```python
    return self.parse_strategy_decision(response)
```

Listing 3: ITRS Zero-Heuristic Approach

This approach enables the system to respond to subtle contextual cues and problem-specific requirements that would be impossible to capture in predetermined rules.

4 Theoretical Analysis

4.1 Convergence Properties

The theoretical foundation of ITRS rests on formal convergence guarantees that ensure the iterative reasoning process terminates at optimal or near-optimal solutions. We define convergence in terms of a composite quality measure that incorporates both solution completeness and confidence levels.

Definition 4.1 (Thought Document Quality). *Let D_t represent the thought document at iteration t. The quality measure $Q(D_t)$ is defined as:*

$$Q(D_t) = \alpha \cdot C(D_t) + \beta \cdot R(D_t) + \gamma \cdot S(D_t) \tag{1}$$

where $C(D_t)$ is the completeness measure, $R(D_t)$ is the consistency measure, $S(D_t)$ is the semantic coherence measure, and $\alpha + \beta + \gamma = 1$ with $\alpha, \beta, \gamma > 0$.

Theorem 4.2 (Convergence Guarantee). *Under the ITRS iterative refinement process, the quality sequence $\{Q(D_t)\}_{t=0}^{\infty}$ is monotonically non-decreasing and bounded above, ensuring convergence to a limit point Q^*.*

Proof. The monotonicity property follows from the design of the refinement strategies, which are constrained to either improve or maintain the quality of each section. The boundedness follows from the finite nature of the solution space for any given problem. Formal convergence then follows from the monotone convergence theorem. □

The practical convergence criterion is defined as:

$$|Q(D_{t+1}) - Q(D_t)| < \epsilon \tag{2}$$

where ϵ is an adaptive threshold determined by the LLM based on problem complexity and required precision.

4.2 Computational Complexity Analysis

The computational complexity of ITRS depends on several factors: the number of thoughts n, the number of iterations k, and the dimensionality of semantic embeddings d.

Proposition 4.3 (Complexity Bounds). *The computational complexity of ITRS is characterized by:*

- *Knowledge graph operations: $O(n^2)$ for relationship detection per iteration*
- *Semantic embedding operations: $O(n \log n)$ with FAISS indexing*
- *Overall system complexity: $O(k \cdot n^2)$ worst case, $O(k \cdot n \log n)$ average case*

The complexity analysis reveals that ITRS scales reasonably with problem size, with the quadratic component dominated by knowledge graph relationship detection, which can be optimized through incremental update strategies.

4.3 Synergistic Effects of Dual Representation

The integration of knowledge graphs and semantic embeddings creates powerful synergistic effects that enhance reasoning capabilities beyond what either approach could achieve independently.

Theorem 4.4 (Representational Completeness). *The combination of discrete symbolic relationships (knowledge graphs) and continuous semantic representations (embeddings) provides representational completeness for the class of reasoning problems addressable by current LLMs.*

This dual representation enables ITRS to capture both logical structure and semantic nuance, providing a comprehensive framework for reasoning that mirrors aspects of human cognitive architecture.

5 Applications and Use Cases

5.1 Explainable AI (XAI) Enhancement

ITRS provides unprecedented capabilities for explainable AI through its structured reasoning traces and iterative refinement process. The system addresses key challenges in XAI:

5.1.1 Progressive Explanation Refinement

Traditional XAI approaches provide static explanations that may be either too simplistic or overwhelming for users. ITRS enables progressive disclosure, starting with high-level explanations and iteratively adding detail based on user needs and feedback.

The system maintains explanations at multiple levels of granularity:

- **Executive Summary**: High-level conclusions and key insights
- **Reasoning Chains**: Step-by-step logical progressions
- **Evidence Base**: Supporting facts and validation results
- **Alternative Perspectives**: Consideration of different viewpoints
- **Uncertainty Assessment**: Detailed confidence mapping and limitation acknowledgment

5.1.2 Contradiction Resolution and Consistency Assurance

The knowledge graph component enables ITRS to detect and resolve contradictions across different parts of the explanation, ensuring coherent and self-consistent explanations. This addresses a critical limitation of current XAI systems that may provide conflicting explanations for related decisions.

5.1.3 Confidence Calibration and Uncertainty Quantification

ITRS provides detailed confidence breakdowns for each component of an explanation, enabling users to understand not just what the system concludes but how certain it is about each aspect of the reasoning. This granular uncertainty quantification is essential for high-stakes applications where understanding system limitations is crucial.

5.2 Trustworthy AI (TAI) Implementation

The ITRS architecture provides comprehensive support for trustworthy AI requirements across multiple dimensions:

5.2.1 Audit Trail and Accountability

Every decision, strategy selection, and refinement step is logged with complete provenance information, creating a comprehensive audit trail. This includes:

- Complete iteration history with timestamps and version tracking
- Strategy selection rationale and parameter optimization decisions
- Knowledge graph evolution and relationship detection
- Semantic embedding analysis and gap identification
- Convergence assessment and termination criteria

This audit trail enables post-hoc analysis of system behavior and supports regulatory compliance requirements in critical applications.

5.2.2 Bias Detection and Mitigation

The iterative refinement process, particularly through the CRITICAL strategy, enables systematic examination of potential biases and assumptions. The system can identify when reasoning has become too narrow or when important perspectives have been overlooked.

The semantic embedding analysis helps detect implicit biases by identifying systematic gaps in the conceptual space being explored, while the knowledge graph reveals potential echo chambers where reasoning reinforces existing assumptions without sufficient critical examination.

5.2.3 Robustness and Reliability Assessment

ITRS provides mechanisms for assessing the robustness of conclusions through:

- Multiple independent reasoning paths leading to the same conclusion
- Systematic stress-testing of assumptions and hypotheses
- Evaluation of sensitivity to parameter variations
- Assessment of reasoning stability across different starting conditions

5.3 General LLM Enhancement

Beyond specialized applications, ITRS serves as a general enhancement layer for existing LLMs, providing capabilities that dramatically improve reasoning quality across diverse tasks.

5.3.1 Complex Problem Decomposition

Multi-faceted queries that would overwhelm traditional single-pass approaches are automatically decomposed through iterative exploration. The system identifies different aspects of complex problems and ensures each receives focused attention while maintaining awareness of interdependencies.

For example, when addressing questions involving multiple domains (such as technical, ethical, and economic considerations), ITRS ensures balanced exploration while identifying connections and trade-offs between different aspects.

5.3.2 Self-Correction and Error Recovery

The VALIDATION and CRITICAL strategies enable models to identify and correct their own errors without requiring external feedback. This self-correction capability is particularly valuable for:

- Detecting logical inconsistencies in reasoning chains
- Identifying unsupported claims or inferential leaps
- Recognizing when initial assumptions may be incorrect
- Recovering from reasoning dead-ends through strategic pivoting

5.3.3 Knowledge Integration and Synthesis

The dynamic knowledge graph construction enables models to maintain consistency across extended reasoning sessions, connecting insights from different iterations and ensuring that new understanding builds coherently on previous insights.

This capability is particularly valuable for research tasks, analytical writing, and complex problem-solving scenarios where maintaining coherent understanding across multiple reasoning sessions is essential.

6 Implementation Details and Technical Considerations

6.1 LLM Decision Making Architecture

The implementation of zero-heuristic decision making requires careful prompt engineering and response parsing to ensure reliable operation while maintaining the emergent nature of decisions.

```python
async def analyze_gaps_comprehensive(
    self,
    doc: ThoughtDocument
) -> Dict[str, Any]:
    """
```

```
    Perform comprehensive gap analysis using LLM intelligence
    to identify areas requiring attention in the current iteration.

    Args:
        doc: Current thought document state

    Returns:
        Structured gap analysis with priorities and recommendations
    """
    analysis_prompt = f"""
    Perform comprehensive gap analysis for iterative reasoning
refinement.

    ORIGINAL QUERY: {doc.query}

    CURRENT DOCUMENT STATE:
    {doc.to_comprehensive_markdown()}

    KNOWLEDGE GRAPH CONTEXT:
    {doc.graph.get_relationship_summary()}

    SEMANTIC ANALYSIS:
    {doc.embeddings.get_coherence_summary()}

    ANALYSIS DIMENSIONS:
    1. QUERY COVERAGE: Is there a direct, complete answer to the
original query?
    2. LOGICAL COMPLETENESS: Are there gaps in reasoning chains?
    3. EVIDENCE SUFFICIENCY: Are claims adequately supported?
    4. PERSPECTIVE BREADTH: Have important viewpoints been considered?
    5. SECTION UTILIZATION: Are all document sections being effectively
    used?
    6. CONSISTENCY CHECK: Are there any contradictions or tensions?
    7. DEPTH ASSESSMENT: Is the analysis sufficiently detailed?
    8. NOVELTY POTENTIAL: Are there unexplored avenues worth
investigating?

    Provide structured analysis with specific, actionable gaps
identified.
    """

    response = await self.llm.generate(
        analysis_prompt,
        temperature=0.3,  # Lower temperature for analytical tasks
        max_tokens=2000
    )

    return self.parse_gap_analysis(response)
```

Listing 4: Gap Analysis Implementation

6.2 Knowledge Graph Implementation

The dynamic knowledge graph construction employs sophisticated relationship detection that goes beyond simple similarity measures:

```
async def detect_relationships_semantic(
    self,
```

```
    new_thought: str,
    existing_thoughts: List[Dict[str, Any]],
    context: ThoughtDocument
) -> List[Relationship]:
    """
    Detect semantic and logical relationships between new thought
    and existing thoughts using LLM analysis combined with
    embedding similarity.

    Args:
        new_thought: Newly added thought content
        existing_thoughts: List of existing thoughts with metadata
        context: Full document context for relationship assessment

    Returns:
        List of detected relationships with types and strengths
    """
    relationships = []

    # Use semantic embeddings for initial filtering
    new_embedding = self.embedding_model.encode(new_thought)
    candidates = self.find_semantic_candidates(
        new_embedding,
        existing_thoughts,
        threshold=0.3
    )

    # Use LLM for precise relationship detection
    for candidate in candidates:
        relationship_prompt = f"""
        Analyze the relationship between these two thoughts:

        NEW THOUGHT: {new_thought}
        EXISTING THOUGHT: {candidate['content']}

        CONTEXT: {context.get_relevant_context(candidate['id'])}

        RELATIONSHIP TYPES:
        - SUPPORTS: Provides evidence or strengthening
        - CONTRADICTS: Direct logical opposition
        - BUILDS_ON: Extension or elaboration
        - ANSWERS: Direct response to question
        - RAISES: Generates new questions
        - REFINES: Improves precision/accuracy
        - CONNECTS: Links disparate concepts
        - VALIDATES: Independent confirmation
        - CHALLENGES: Questions assumptions
        - EXPLAINS: Provides causal mechanism
        - REQUIRES: Indicates dependency

        If a relationship exists, specify type and strength (0.0-1.0).
        If no meaningful relationship, respond "NONE".
        """

        response = await self.llm.generate(relationship_prompt)
        relationship = self.parse_relationship_response(response)

        if relationship:
```

```python
            relationships.append(relationship)

    return relationships
```

Listing 5: Relationship Detection Implementation

6.3 Semantic Vector Engine Implementation

The semantic vector engine integrates advanced embedding analysis with clustering and gap detection:

```python
async def detect_semantic_gaps(
    self,
    query_embedding: np.ndarray,
    thought_embeddings: List[np.ndarray],
    coverage_threshold: float = 0.7
) -> List[SemanticGap]:
    """
    Detect gaps in semantic coverage using embedding space analysis
    and clustering techniques.

    Args:
        query_embedding: Original query embedding
        thought_embeddings: List of all current thought embeddings
        coverage_threshold: Minimum coverage for completeness

    Returns:
        List of identified semantic gaps with exploration suggestions
    """
    # Perform HDBSCAN clustering to identify semantic neighborhoods
    clusterer = hdbscan.HDBSCAN(
        min_cluster_size=2,
        metric='cosine',
        cluster_selection_epsilon=0.3
    )

    cluster_labels = clusterer.fit_predict(thought_embeddings)

    # Identify uncovered regions in embedding space
    gaps = []

    # Check for regions between clusters
    cluster_centers = self.calculate_cluster_centers(
        thought_embeddings,
        cluster_labels
    )

    # Use query embedding as reference point
    query_distances = cosine_distances(
        [query_embedding],
        cluster_centers
    )[0]

    # Identify potential gaps based on distance analysis
    for i, distance in enumerate(query_distances):
        if distance > coverage_threshold:
            gap_direction = self.calculate_gap_direction(
                query_embedding,
```

```
                cluster_centers[i]
            )

            gap = SemanticGap(
                direction=gap_direction,
                distance=distance,
                cluster_id=i,
                exploration_suggestion=await self.
generate_exploration_suggestion(
                    gap_direction,
                    query_embedding
                )
            )
            gaps.append(gap)

    return gaps
```

Listing 6: Semantic Gap Detection

6.4 Dynamic Parameter Optimization

The elimination of hardcoded parameters requires sophisticated dynamic optimization:

```
async def optimize_parameters_dynamic(
    self,
    strategy: RefinementStrategy,
    doc: ThoughtDocument,
    gaps: Dict[str, Any],
    iteration: int
) -> Dict[str, float]:
    """
    Dynamically optimize parameters for current refinement iteration
    using LLM intelligence and context analysis.

    Args:
        strategy: Selected refinement strategy
        doc: Current document state
        gaps: Identified gaps from analysis
        iteration: Current iteration number

    Returns:
        Optimized parameters for current iteration
    """
    optimization_prompt = f"""
    Optimize parameters for iterative reasoning refinement.

    STRATEGY: {strategy.value}
    ITERATION: {iteration}

    DOCUMENT STATE:
    - Total sections: {len(doc.sections)}
    - Average confidence: {doc.get_average_confidence():.3f}
    - Complexity score: {doc.calculate_complexity_score():.3f}

    IDENTIFIED GAPS:
    {self.format_gaps_for_optimization(gaps)}

    PARAMETER OPTIMIZATION:
```

```
    1. TEMPERATURE: Balance between creativity and precision
       - Higher for exploration/creativity (0.7-0.9)
       - Lower for validation/precision (0.1-0.4)
       - Consider current confidence levels and strategy needs

    2. MAX_TOKENS: Appropriate response length
       - Consider section complexity and gaps identified
       - Balance thoroughness with efficiency

    3. TOP_P: Nucleus sampling parameter
       - Adjust based on desired output diversity
       - Consider strategy requirements

    4. FOCUS_DEPTH: How deeply to explore identified gaps
       - Scale: 1.0 (surface) to 3.0 (deep dive)
       - Consider iteration budget and urgency

    Provide specific numeric values with brief justification.
    """

    response = await self.llm.generate(optimization_prompt)
    return self.parse_parameter_optimization(response)
```

Listing 7: Dynamic Parameter Optimization

7 Framework for Future Empirical Validation

While the theoretical foundations and architectural components of ITRS have been thoroughly developed, comprehensive empirical validation remains an important direction for future work. This section outlines the proposed framework for systematic evaluation of ITRS performance across multiple dimensions.

7.1 Proposed Evaluation Methodology

Future empirical studies should assess ITRS effectiveness across three primary dimensions:

1. **Reasoning Quality Assessment**: Evaluation through expert analysis, logical consistency verification, and solution completeness measures across diverse reasoning domains
2. **Transparency and Explainability Evaluation**: User studies assessing explanation quality, comprehensibility, and trustworthiness with domain experts and end users
3. **Computational Efficiency Analysis**: Systematic measurement of convergence properties, resource utilization, and scalability characteristics

7.2 Proposed Benchmark Categories

The comprehensive evaluation framework should encompass four categories of reasoning tasks:

7.2.1 Logical Reasoning Tasks

Complex multi-step logical problems requiring sustained reasoning chains, including formal logic puzzles, mathematical proofs, and analytical reasoning challenges that test the system's ability to maintain logical consistency across iterations.

7.2.2 Scientific Analysis Tasks

Problems requiring integration of multiple scientific domains, hypothesis generation and testing, and synthesis of potentially contradictory evidence to evaluate the system's capability for evidence-based reasoning.

7.2.3 Ethical Dilemma Analysis

Complex ethical scenarios requiring consideration of multiple stakeholder perspectives, value conflicts, and consequential analysis to assess the system's ability to handle nuanced moral reasoning.

7.2.4 Strategic Planning Tasks

Business and policy planning scenarios requiring consideration of multiple objectives, constraint analysis, and stakeholder impact assessment to evaluate complex decision-making capabilities.

7.3 Comparative Baseline Framework

Future empirical studies should compare ITRS performance against established state-of-the-art baselines:

- **Chain-of-Thought Prompting**: Traditional CoT approaches [Wei et al., 2022] to establish baseline reasoning capabilities
- **Tree-of-Thoughts**: Structured exploration methods [Yao et al., 2023] to compare against systematic reasoning approaches
- **Graph-of-Thoughts**: Graph-based reasoning with aggregation [Besta et al., 2024a] to evaluate against structured knowledge representation
- **Self-Refine**: Iterative self-improvement approaches [Madaan et al., 2023] to compare iterative refinement strategies

7.4 Evaluation Metrics and Success Criteria

The proposed evaluation framework should incorporate both quantitative and qualitative metrics:

7.4.1 Quantitative Metrics

- Solution accuracy and completeness scores
- Convergence rates and iteration efficiency
- Computational resource utilization
- Logical consistency measures across reasoning chains
- Knowledge graph connectivity and relationship quality

7.4.2 Qualitative Metrics

- Expert assessment of reasoning quality and sophistication
- User studies evaluating explanation comprehensibility
- Trust and confidence ratings from domain experts
- Assessment of transparency and auditability features
- Evaluation of practical applicability across domains

8 Limitations and Future Directions

8.1 Current Limitations

Despite its significant advances, ITRS has several limitations that present opportunities for future research:

8.1.1 Computational Overhead

The iterative nature of ITRS introduces computational overhead that may be excessive for simple queries. The system's multi-iteration approach, while beneficial for complex reasoning tasks, may be inefficient for straightforward problems that could be solved adequately with single-pass inference.

8.1.2 Dependence on Base LLM Quality

The effectiveness of ITRS is fundamentally limited by the capabilities of the underlying LLM. While the architecture enhances reasoning through iterative refinement, it cannot overcome fundamental limitations in the base model's knowledge or reasoning capabilities.

8.1.3 Potential for Over-refinement

In some cases, ITRS may continue refining solutions that are already adequate, leading to diminishing returns and unnecessary computational expense. The current convergence criteria, while adaptive, may not always capture the optimal stopping point.

8.1.4 Scalability Challenges

As the number of thoughts and relationships grows, the knowledge graph operations become computationally expensive. While the average complexity is manageable, worst-case scenarios with dense thought connectivity can lead to significant performance degradation.

8.2 Future Research Directions

Several promising directions for future research emerge from the current work:

8.2.1 Adaptive Iteration Budgets

Developing sophisticated mechanisms for predicting optimal iteration budgets based on query complexity, domain characteristics, and quality requirements. This could involve machine learning approaches to predict convergence behavior and optimize computational resource allocation.

8.2.2 Multi-Agent ITRS Architecture

Extending ITRS to a multi-agent framework where specialized reasoning agents focus on different aspects of complex problems. This could enable parallel processing while maintaining the coherent integration that characterizes single-agent ITRS.

8.2.3 External Knowledge Integration

Incorporating external knowledge bases and real-time information retrieval into the ITRS reasoning loop. This would enable the system to ground its reasoning in up-to-date factual knowledge while maintaining the iterative refinement benefits.

8.2.4 Hardware Acceleration

Developing specialized hardware architectures optimized for the knowledge graph operations and semantic embedding computations that are central to ITRS performance.

8.2.5 Formal Verification Integration

Incorporating formal verification techniques to provide mathematical guarantees about reasoning correctness in critical applications.

8.3 Broader Implications

The ITRS architecture has several broader implications for the field of AI reasoning:

8.3.1 Emergent Intelligence Paradigm

The zero-heuristic approach pioneered in ITRS suggests a broader paradigm shift toward emergent intelligence systems where AI components make decisions traditionally handled by hardcoded rules. This approach may be applicable to many other AI system architectures.

8.3.2 Transparency and Accountability Standards

The comprehensive audit trails and reasoning traces provided by ITRS establish new standards for transparency in AI systems. As AI deployment in critical applications increases, such transparency mechanisms may become regulatory requirements.

8.3.3 Human-AI Collaboration

The structured reasoning process and progressive explanation capabilities of ITRS enable new forms of human-AI collaboration where humans can interact with and guide the reasoning process at multiple levels of detail.

9 Conclusion

This paper has presented the Iterative Thought Refinement System (ITRS), a novel architecture that represents a fundamental advancement in AI reasoning systems. Through the elimination of hardcoded heuristics in favor of emergent LLM intelligence, integration of dynamic knowledge graphs with semantic embeddings, and provision of complete transparency through structured reasoning traces, ITRS addresses critical limitations in current reasoning approaches.

The key contributions of this work include:

1. The first fully LLM-driven iterative reasoning architecture with zero-heuristic decision making

2. A novel approach to dynamic knowledge graph construction that captures reasoning evolution rather than static knowledge

3. Integration of semantic embeddings for active reasoning guidance beyond simple retrieval

4. Formal convergence guarantees and computational complexity analysis

5. Comprehensive applications demonstrating effectiveness across explainable AI, trustworthy AI, and general LLM enhancement

Our theoretical analysis demonstrates the potential for significant improvements in reasoning quality, transparency, and reliability across diverse task domains. The system's formal convergence guarantees and computational complexity bounds provide a solid foundation for future empirical validation.

The broader implications of ITRS extend beyond technical improvements to suggest new paradigms for AI system design. The elimination of hardcoded heuristics in favor of emergent intelligence represents a fundamental shift that may influence future AI architectures across multiple domains. The comprehensive transparency and audit trail capabilities establish new standards for accountable AI systems.

As AI systems take on increasingly complex reasoning tasks in critical applications, architectures like ITRS that combine iterative refinement, structured knowledge representation, and semantic understanding will become essential. The zero-heuristic approach demonstrates that AI systems can be both more capable and more transparent through thoughtful architectural design.

Future research directions include adaptive iteration budgets, multi-agent architectures, external knowledge integration, and hardware acceleration. The foundation established by ITRS provides a robust platform for continued advancement in AI reasoning systems.

The development of ITRS represents a significant step toward AI systems that can engage in sophisticated, transparent, and reliable reasoning while maintaining the adaptability and context-awareness that are hallmarks of intelligent behavior. As we continue to push the boundaries of what AI systems can achieve, architectures like ITRS will play a crucial role in ensuring that increased capability is accompanied by increased transparency and trustworthiness.

References

Aws Albarghouthi. Introduction to neural network verification. *Foundations and Trends in Programming Languages*, 7(1-2):1–157, 2021. arXiv:2109.10317.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, et al. Constitutional ai: Harmlessness from ai feedback. arXiv preprint arXiv:2212.08073, 2022.

Maciej Besta, Nils Blach, Aleš Kubíček, Robert Gerstenberger, Michał Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the 38th AAAI Conference on Artificial Intelligence*, 2024a. arXiv:2308.09687.

Maciej Besta, Florim Memedi, Zhenyu Zhang, Robert Gerstenberger, et al. Demystifying chains, trees, and graphs of thoughts. arXiv preprint arXiv:2401.14295, 2024b.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. In *Advances in Neural Information Processing Systems*, 2024. arXiv:2405.14831.

Bernal Jiménez Gutiérrez, Yiheng Shu, Weijian Qi, Sizhe Zhou, and Yu Su. From rag to memory: Non-parametric continual learning for large language models. arXiv preprint arXiv:2502.14802, 2025.

Donghoon Han, Eunhwan Park, Gisang Lee, Adam Lee, and Nojun Kwak. Merlin: Multimodal embedding refinement via llm-based iterative navigation for text-video retrieval-rerank pipeline. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 532–544, 2024.

Xiaoxin He, Yijun Tian, Yifei Wang, Nitesh V. Chawla, Laurent Itti, and Xiangliang Zhang. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. In *Advances in Neural Information Processing Systems*, 2024. arXiv:2402.07630.

Bowen Jin, Chulin Xie, Jiawei Zhang, et al. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 165–186, 2024.

Guy Katz, Clark Barrett, David L. Dill, Kyle Julian, and Mykel J. Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *Computer Aided Verification*, pages 97–117. Springer, 2017. arXiv:1702.01135.

Guy Katz, Derek A. Huang, Duligur Ibeling, et al. The marabou framework for verification and analysis of deep neural networks. In *Computer Aided Verification*, pages 443–452. Springer, 2019.

Dominik Kowald, Sebastian Scher, Viktoria Pammer-Schindler, et al. Establishing and evaluating trustworthy ai: Overview and research challenges. *Frontiers in Big Data*, 7:1467222, 2024.

Bo Li, Peng Qi, Bo Liu, Shuai Di, et al. Trustworthy ai: From principles to practices. *ACM Computing Surveys*, 56(2):1–46, 2023. arXiv:2110.01167.

Yunfei Li, Bowen Wu, Yingkai Huang, and Shaoxuan Luan. Developing trustworthy artificial intelligence: Insights from research on interpersonal, human-automation, and human-ai trust. *Frontiers in Psychology*, 15:1382693, 2024.

Weixin Liang, Girmaw Abebe Tadesse, Daniel Ho, Li Fei-Fei, Matei Zaharia, Ce Zhang, and James Zou. Advances, challenges and opportunities in creating data for trustworthy ai. *Nature Machine Intelligence*, 4(8):669–677, 2022.

Haochen Liu, Song Wang, Yaochen Zhu, Yushun Dong, and Jundong Li. Knowledge graph-enhanced large language models via path selection. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6287–6301, 2024.

Diego Manzanas Lopez, Sung Woo Choi, Hoang-Dung Tran, and Taylor T. Johnson. Nnv 2.0: The neural network verification tool. In *Computer Aided Verification*, pages 397–412. Springer, 2023.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, et al. Self-refine: Iterative refinement with self-feedback. arXiv preprint arXiv:2303.17651, 2023.

Elan Markowitz, Anil Ramakrishna, Jwala Dhamala, Ninareh Mehrabi, Charith Peris, Rahul Gupta, Kai-Wei Chang, and Aram Galstyan. Tree-of-traversals: A zero-shot reasoning algorithm for augmenting black-box language models with knowledge graphs. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, pages 11998–12014, 2024.

Shirui Pan, Linhao Luo, Yufei Wang, Chen Chen, Jiapu Wang, and Xindong Wu. Unifying large language models and knowledge graphs: A roadmap. arXiv preprint arXiv:2306.08302, 2023.

Jayneel Parekh, Pegah Khayatan, Mustafa Shukor, Alasdair Newson, and Matthieu Cord. A concept-based explainability framework for large multimodal models. In *Advances in Neural Information Processing Systems*, 2024. arXiv:2406.08074.

Ahmed Salih, Kjersti Engan, Joana Figueira, et al. A perspective on explainable artificial intelligence methods: Shap and lime. arXiv preprint arXiv:2305.02012, 2023.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. arXiv preprint arXiv:2203.11171, 2022.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*, 2022. arXiv:2201.11903.

Weimin Xiong, Yifan Song, Xiutian Zhao, et al. Watch every step! llm agent learning via iterative step-level process refinement. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1576–1602, 2024.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, 2023. arXiv:2305.10601.