From Chess Intuition to AI Innovation: A Concept for Automated Creative Problem-Solving

Abstract

The codification of human intuition and creative problem-solving represents a fundamental challenge in artificial intelligence research. Contemporary AI systems demonstrate exceptional performance in structured, logic-based reasoning tasks, however, their capacity to emulate human-like lateral thinking and intuitive insight remains significantly constrained.

The chess grandmaster's ability to instantly evaluate positions through pattern recognition. Being able to process millions of learned configurations to generate intuitive judgments in milliseconds. This provides a concrete computational model for encoding expert intuition. This paper demonstrates how the hierarchical pattern libraries, evaluation heuristics, and position-based reasoning that characterize chess expertise can be systematically translated into the IIA framework's architecture, creating a rigorous bridge between human intuitive expertise and AI-driven innovation.

This paper presents a conceptual framework for an Intuitive Inventive Agent (IIA) that bridges this gap by integrating the Theory of Inventive Problem Solving (TRIZ) with lateral thinking methodologies, operationalized through state-of-the-art Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), and orchestration architectures employing LangGraph and n8n.

This approach synthesizes three key innovations:

(1) Automation of TRIZ's systematic contradiction resolution through LLM-powered semantic analysis and deterministic principle retrieval,

(2) Incorporation of Edward de Bono's lateral thinking techniques to expand solution spaces beyond conventional boundaries, and

(3) Implementation of an Agent orchestration architecture that coordinates multi-agent workflows while maintaining transparency and human interpretability. The framework leverages insights from competitive chess expertise to model rapid, experience-driven judgment patterns characteristic of expert intuition, addressing the distinction between Kahneman's System 1 (fast, intuitive) and System 2 (slow, analytical) cognitive processes.

The IIA concept framework advances the state-of-the-art by providing the first unified automation of structured inventive problem-solving and creative lateral thinking within a single computational architecture. This work contributes to the broader goal of developing AI systems that not only process information but demonstrate wisdom. This is achieved by the integration of knowledge, experience, and contextual judgment that characterizes human expertise.

Keywords: Theory of Inventive Problem Solving (TRIZ), lateral thinking, human intuition codification, Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), multi-agent orchestration, LangGraph, n8n, automated innovation, cognitive architecture, System 1/System 2 cognition, AlphaEvolve, DeepResearch Agents.

1. Introduction

The pursuit of artificial intelligence systems capable of genuine innovation represents one of the most profound challenges in computational research. Large Language Models (LLMs) have achieved remarkable proficiency in structured reasoning tasks and demonstrate capabilities in formal logic, mathematical proof, and multi-step analytical processes. Yet there remains a gap in replicating the intuitive leaps and creative insights that characterize human problem-solving expertise. This fundamental limitation, which we term the "human intuition gap," prevents current AI systems from achieving the kind of breakthrough innovations that emerge from the synthesis of analytical rigor and experiential wisdom.

A new field of Custom DeepResearch Agents (e.g. Google, LangChain etc) is evolving fast, this approach intends to supplement these DeepResearch agents, with guide-rails and optimal orchestration flows.

1.1 The Nature of Intuitive Innovation

Innovation in technical domains emerges from a paradoxical cognitive process: it requires both systematic analysis and creative intuition. Kahneman's dual-process theory provides a useful framework for understanding this dichotomy, distinguishing between System-2 cognition (deliberate, analytical reasoning) and System-1 cognition (fast, intuitive, pattern-based thinking) [1]. Current AI architectures excel at System-2 processes but struggle to replicate System-1's rapid, experience-driven insights that often lead to breakthrough innovations.

The Theory of Inventive Problem Solving (TRIZ), developed by Genrich Altshuller through analysis of millions of patents, offers a systematic approach to innovation by identifying and resolving technical contradictions. These are the situations where improving one system parameter necessarily degrades another. However, even TRIZ's structured methodology relies heavily on human intuition for problem framing, principle selection, and solution adaptation. This dependency highlights a critical challenge: how can we encode the tacit knowledge and intuitive judgment that experts develop through years of practice into computational systems?

1.2 Research Motivation and Questions

This research is motivated by a unique synthesis of expertise spanning competitive strategy, systematic innovation, and AI, providing a distinct perspective on codifying human intuition. My approach is grounded in three core domains:

Expert Intuition from Competitive Chess: The primary insight stems from over 35,000 hours of study and chess-competitions, culminating in Lichess.org 2700 Elo rating. In high-level chess, the ability to instantly "feel" the strength of a position without exhaustive calculation is paramount. This is a living laboratory for Kahneman's System-1 thinking, a rapid, pattern-based judgment honed through experience. While LLMs possess vast intelligence, this research aims to imbue them with this form of wisdom, which arises from the synthesis of knowledge and deeply learned patterns.

Structured Innovation Methodologies: This intuitive foundation is balanced with formal training in rigorous, analytical frameworks. Certification in Oxford TRIZ provides a deep understanding of systematic problem-solving, contradiction analysis, and the 40 Inventive Principles. This is complemented by extensive practice in Edward de Bono's lateral thinking techniques (e.g., Six Thinking Hats, Concept Extraction), which provide structured methods for breaking cognitive fixedness and exploring unconventional solutions.

Technical and Scientific Execution: The vision to merge these domains is made concrete through a strong foundation in machine learning and data science, including an MIT MicroMasters in Statistics and Data Science. This theoretical knowledge is paired with hands-on experience in building complex systems using Python, LangChain, and modern agentic orchestration tools, ensuring that the proposed framework is not just conceptually sound but technically feasible.

This convergence of firsthand experience in intuition (chess), formal training in innovation (TRIZ, de Bono), and technical expertise (AI/ML) provides a unique vantage point from which to address the core challenges of creating truly inventive AI. Consequently, this research is guided by the following fundamental questions:

RQ1: Architectural Integration - How can we design a computational architecture based on Agentic AI that seamlessly integrates structured innovation methodologies (TRIZ) with creative lateral thinking frameworks within modern LLM-based systems, enabling both systematic contradiction resolution and exploratory ideation?

RQ2: Cognitive Simulation - What architectural patterns and orchestration mechanisms enable AI systems to effectively simulate both analytical (System 2) and intuitive (System 1) cognitive processes, particularly in navigating the trade-offs inherent in inventive problem-solving?

RQ3: Intuition Quantification - How can we develop robust, reproducible metrics to evaluate the "intuitive quality" of AI-generated solutions, distinguishing between mere randomness and the kind of informed creativity that characterizes expert human judgment?

1.3 Contributions and Approach

This paper presents the Intuitive Inventive Agent (IIA), a novel framework that bridges the gap between analytical reasoning and creative intuition in AI-driven innovation. The approach makes four primary contributions:

Theoretical Contribution: formalize a computational model for encoding intuitive reasoning patterns derived from expert problem-solving behaviors, grounded in cognitive science principles and validated through empirical analysis of human innovation processes. This model provides a theoretical foundation for understanding how experiential wisdom can be translated into algorithmic form.

Architectural Innovation: RAG-Graph based AgenticAI Orchestration architecture that maintains semantic relationships between innovation principles while enabling dynamic knowledge retrieval and contextual reasoning. This architecture, orchestrated through a dual-layer system combining LangGraph for internal cognitive processes and n8n for external system integration, enables unprecedented coordination between analytical and creative reasoning modules.

Methodological Synthesis: successful integration of TRIZ's systematic contradiction resolution with de Bono's lateral thinking techniques within a unified computational framework. This synthesis enables AI systems to navigate both the structured search spaces of technical innovation and the open-ended exploration required for creative breakthroughs.

Empirical Validation: comprehensive evaluation metrics and benchmarks specifically designed for assessing creative AI systems, addressing the long-standing challenge of quantifying intuitive quality in computational outputs. Our evaluation framework considers novelty, feasibility, contextual relevance, and the presence of non-obvious insights that characterize human expertise.

2. Related Work and Background

2.1 TRIZ: Structured Inventive Problem Solving

The Theory of Inventive Problem Solving (TRIZ), developed by Genrich Altshuller, is a systematic methodology built on the premise that innovation follows predictable patterns. Its core tenet is that inventive solutions emerge from resolving system contradictions rather than accepting compromises. While TRIZ is a comprehensive philosophy with numerous tools, our research focuses on operationalizing its central, deterministic workflow to create a computationally tractable engine for innovation.

The components of TRIZ automated within our Intuitive Inventive Agent (IIA) framework are:

Contradiction Analysis: The process begins by framing a problem as a Technical Contradiction. This occurs when an attempt to improve one desirable feature of a system, known as an Improving Parameter, leads to the degradation of another, the Worsening Parameter. TRIZ defines 39 such universal engineering parameters (e.g., Weight, Speed, Reliability), providing a standardized language for problem definition.

The Contradiction Matrix: The cornerstone of classical TRIZ is the 39×39 Contradiction Matrix. This deterministic tool serves as a knowledge base, mapping a specific conflict between an improving and a worsening parameter to a small, statistically-proven set of inventive principles most likely to resolve it. This lookup mechanism is ideally suited for automation.

The 40 Inventive Principles: The outputs of the matrix are drawn from a set of 40 Inventive Principles. These are abstract, universal strategies for problem-solving derived from the analysis of millions of patents (e.g., Segmentation, Asymmetry, Phase Transitions). In our framework, these principles are not the final answer but serve as powerful, context-rich prompts for the LLM to generate specific, actionable solutions.

ARIZ-Inspired Orchestration: The overall process flow is guided by the principles of the Algorithm of Inventive Problem Solving (ARIZ). While a full ARIZ implementation is a complex, multi-stage process, our framework adopts its core logic: systematically defining the problem, identifying the core contradiction, retrieving solution principles, and analyzing the results.

2.2 LLMs in TRIZ Automation

LLMs have demonstrated potential in TRIZ-related automation. Tools such as AutoTRIZ and TRIZ-GPT execute steps from contradiction identification to principle-based solution synthesis [2][3]. Multi-agent approaches simulate the dynamics of expert innovation teams [4].

Challenges remain:

- Hallucination and factual errors
- Limited evaluation metrics for creativity
- Dependency on human interpretation
- Difficulty in generalizing to novel problem spaces

2.3 Lateral Thinking and Human Intuition in AI

Edward de Bono's lateral thinking frameworks encourage idea generation through the disruption of conventional patterns, reframing, and exploration of improbable alternatives.

Emerging AI research has begun to incorporate aspects of this approach: SPLAT for non-obvious puzzles, DCoLT for non-linear reasoning, and SALT for agentic creative processes [5][6][7]. These developments provide a foundation for simulating System 1 cognition within AI systems.

3. Proposed Framework: Intuitive Inventive Agent

The Intuitive Inventive Agent (IIA) framework is designed to integrate structured inventive problem-solving methodologies with creative, intuitive reasoning in a unified computational architecture. The framework integrates:

- TRIZ methodology for structured problem resolution
- Lateral thinking tools for creative exploration
- LLMs for Intelligence
- Google Search & Custom Knowledge base for World Knowledge
- RAG Graph : To Create a World View in Embedding format withInterconnections between various elements.
- LangChain and n8n for orchestration and automation of the Thinking process.

3.1 LLM-Powered TRIZ Module

The TRIZ pipeline is operationalized through four orchestrated stages:

- Problem clarification via LLM-driven refinement
- Contradiction detection using the 39 TRIZ engineering parameters
- Deterministic retrieval of inventive principles from the Contradiction Matrix
- Solution synthesis grounded in RAG-Graph-fed TRIZ logic

3.2 Lateral Thinking Module

This module simulates Edward de Bono's methods through structured prompts:

- Concept Extraction
- PMI (Plus/Minus/Interesting)
- CAF (Consider All Factors)
- Assumption-Challenging
- Provocations (Po statements)
- Random Word Generation

Outputs are integrated back into TRIZ synthesis to expand the solution search space.

3.3 Orchestration with n8n & LangGraph

To operationalize the multi-step reasoning process and coordinate various agents effectively, the system leverages both n8n and LangGraph. These tools serve as the backbone of orchestration, allowing complex workflows to be visualized, managed, and executed reliably.

n8n acts as the external orchestrator, managing API calls, data ingestion from external tools (like Google Search or Wikipedia), and flow control between components such as the LLM, knowledge graph, and user interface. Its visual programming interface also makes it easier to review and debug the entire pipeline, ensuring transparency and adaptability.

LangGraph, on the other hand, provides a native orchestration layer for coordinating thought processes *within* the language model ecosystem. It allows for the creation of persistent memory across interactions, agent role-switching, and iterative refinement loops. For instance, LangGraph can control when to ask clarifying questions, when to apply TRIZ logic, or when to introduce lateral thinking provocations based on the current state of the problem.

Together, n8n and LangGraph complement each other: n8n ensures smooth integration with external systems, while LangGraph enables structured, multi-turn thinking inside the AI itself. The combined orchestration allows the system to simulate a full cognitive process from data acquisition to structured reasoning while enabling human-in-the-loop checkpoints, real-time monitoring, and modular extensibility.

The Orchestration Flows are inspired by Google AlphaEvolve approach[9]. AlphaEvolve's functionality centers on creating **Evolutionary Agents** that emulate human problem-solving by combining a creative Large Language Model (LLM) with a logical evolutionary algorithm. This process mimics human thinking by using the LLM as a "generator" to brainstorm a diverse population of code variations, much like a human would explore different potential solutions. A "selector" then acts as the critical thinking component, systematically testing each solution against a defined goal or "fitness function" (e.g., speed or correctness) and discarding the failures. This cycle of creative generation followed by rigorous selection allows the agent to iteratively refine solutions, discovering novel approaches that might not be immediately obvious to a human programmer, effectively mirroring an accelerated, focused version of human innovation.

A Unique approach here is to Diverge-Converge based on Design-Thinking Principles. Initially the middle-section of the Orchestration Flow will create a lot of Seed-Potential-Ideas based on TRIZ. These Ideas will then be Ranked against each other using a Play-Off-Tournament like Chess. The ELO of each idea will be evaluated, and finally we will have the top 10 Ideas bubbled up to be selected from.

Playoff Evaluation Framework Summary

The Playoff Evaluation Framework comparative evaluation system designed to rigorously assess and rank multiple ideas, strategies, or solutions through structured "play-off matches." At its core, the framework consists of two specialized agents working in tandem: the Play-Off-Evaluation Framework Creation Agent and the Idea Evaluator & Play-off Agent.

The Play-Off-Evaluation Framework Creation Agent serves as the architect of assessment criteria, researching and identifying domain-specific best practices and evaluation frameworks tailored to each use case. For management consulting, it might establish criteria based on Porter's Five Forces, market attractiveness, and resource alignment. For patent research, it defines standards around USPTO guidelines, claim clarity, and differentiation from prior art. For product innovation, it creates metrics encompassing market demand, technical feasibility, user desirability, and brand alignment. This agent ensures that all evaluations are grounded in industry-standard methodologies while defining clear, objective, and measurable KPIs. It structures "Well Architected Assessment" rules that guarantee fairness, transparency, and strategic alignment, identifying the KPIs that matter most to the specific industry, product, or domain.

Once the framework is established, the Idea Evaluator & Play-off Agent executes the comparative evaluations using these criteria. This agent conducts systematic "league matches" between competing ideas, applying both the custom evaluation framework and established analytical tools like De Bono's thinking methods (PMI, CAF, etc.) and Multi-Criteria Decision Analysis (MCDM) algorithms. The system is designed to handle the massive scale of idea generation produced by the TRIZ-x-DATT Matrix, which generates a minimum of 2,400 seed ideas through the systematic combination of TRIZ's 40 principles with De Bono's thinking tools. This playoff system ensures that from thousands of potential solutions, only the most robust, innovative, and strategically aligned ideas advance to implementation, providing organizations with a rigorous, transparent, and scalable method for innovation management.

Over a period of time, the system will learn about various domains, and create an “Evolutionary” architecture which will strengthen various paths of “Ideas”, and strengthen the “Neuron-Circuits” of the Ideation framework. This will help to build rapid-intuition for situations/cases where deep-thinking is not required.

3.4 Proposed Algorithms

The Intuitive Inventive Agent (IIA) framework proposes several core algorithms that orchestrate the integration of TRIZ methodology, lateral thinking techniques, and LLM-based reasoning. This section presents the fundamental algorithms that power the system's cognitive processes, from problem analysis to solution synthesis.

3.4.1 Master Orchestration Algorithm

The master orchestration algorithm coordinates the entire problem-solving pipeline, managing the flow between analytical and creative modules while maintaining state consistency across multi-agent interactions.

Algorithm: MasterOrchestration

```
Input: problem_statement, context, user_constraints
Output: solution_set, reasoning_trace

1: Initialize:
   knowledge_graph ← LoadKnowledgeGraph()
   triz_matrix ← LoadTRIZContradictionMatrix()
   lateral_tools ← InitializeLateralThinkingTools()
   llm_ensemble ← InitializeLLMAgents()

2: // Problem Understanding Phase
   refined_problem ← ProblemRefinement(problem_statement, context)
   problem_vector ← GenerateEmbedding(refined_problem)

3: // Dual-Path Processing
   parallel:
      analytical_path ← TRIZPipeline(refined_problem, triz_matrix)
      creative_path ← LateralThinkingPipeline(refined_problem, lateral_tools)

4: // Solution Integration
   candidate_solutions ← MergeSolutions(analytical_path, creative_path)

5: // Evolutionary Refinement (AlphaEvolve-inspired)
   for iteration in 1 to MAX_ITERATIONS:
      evolved_solutions ← EvolutionaryRefinement(candidate_solutions)
      if ConvergenceCriteria(evolved_solutions):
         break

6: // Validation and Ranking
   validated_solutions ← ValidateSolutions(evolved_solutions, user_constraints)
```

```
  solution_set ← RankByIntuitionMetrics(validated_solutions)

7: return solution_set, reasoning_trace
```

3.4.2 TRIZ Contradiction Resolution Algorithm

This algorithm implements the systematic TRIZ methodology for identifying and resolving technical contradictions through the application of inventive principles.

Algorithm: TRIZPipeline

```
Input: problem_description, contradiction_matrix
Output: triz_solutions, contradiction_analysis

1: // Contradiction Identification
  parameters ← ExtractEngineeringParameters(problem_description)
  contradictions ← []

2: for each param_pair in parameters:
    if DetectContradiction(param_pair.improving, param_pair.worsening):
      contradiction ← {
        'improving': MapToTRIZParameter(param_pair.improving),
        'worsening': MapToTRIZParameter(param_pair.worsening)
      }
      contradictions.append(contradiction)

3: // Principle Retrieval
  inventive_principles ← []
  for each contradiction in contradictions:
    principles ← contradiction_matrix[contradiction.improving][contradiction.worsening]
    inventive_principles.extend(principles)

4: // Solution Generation
  triz_solutions ← []
  for each principle in inventive_principles:
    principle_description ← RetrievePrincipleDetails(principle)

    // LLM-based contextualization
    contextualized_solution ← LLMGenerate(
      prompt="Apply TRIZ principle: {principle_description} to problem:
```

```
{problem_description}",
        temperature=0.3  // Lower temperature for analytical reasoning
      )

      // RAG-enhanced solution refinement
      similar_cases ← RAGRetrieve(contextualized_solution, knowledge_graph)
      refined_solution ← RefineWithCases(contextualized_solution, similar_cases)

      triz_solutions.append(refined_solution)

5: // Apply 7 Pillars validation
   for each solution in triz_solutions:
      solution.ideality_score ← AssessIdeality(solution)
      solution.resources ← IdentifyResources(solution)
      solution.contradictions_resolved ← ValidateContradictionResolution(solution)

6: return triz_solutions, contradictions
```

3.4.3 Lateral Thinking Generation Algorithm

This algorithm implements Edward de Bono's lateral thinking techniques to generate creative, non-obvious solutions that complement the structured TRIZ approach.

Algorithm: LateralThinkingPipeline

```
Input: problem_description, lateral_tools
Output: creative_solutions

1: creative_solutions ← []
   thinking_modes ← ['white', 'red', 'black', 'yellow', 'green', 'blue']

2: // Six Thinking Hats Analysis
   for each mode in thinking_modes:
      perspective ← ApplySixThinkingHats(problem_description, mode)
      insights ← ExtractInsights(perspective)
      creative_solutions.extend(GenerateFromInsights(insights))

3: // Concept Extraction
   core_concepts ← ExtractConcepts(problem_description)
   for each concept in core_concepts:
```

```
    alternative_concepts ← FindAnalogies(concept, knowledge_graph)
    cross_domain_solutions ← CrossPollinateConcepts(concept, alternative_concepts)
    creative_solutions.extend(cross_domain_solutions)

4: // Provocative Operations (Po)
  provocations ← GenerateProvocations(problem_description)
  for each provocation in provocations:
    po_statement ← "Po: " + provocation
    movement ← ExtractMovement(po_statement, problem_description)
    creative_solutions.append(movement)

5: // Random Word Stimulation
  random_words ← SelectRandomWords(n=5)
  for each word in random_words:
    associations ← GenerateAssociations(word, problem_description)
    stimulated_ideas ← SynthesizeIdeas(associations, problem_description)
    creative_solutions.extend(stimulated_ideas)

6: // PMI Analysis
  for each solution in creative_solutions:
    solution.plus_points ← IdentifyPositives(solution)
    solution.minus_points ← IdentifyNegatives(solution)
    solution.interesting_points ← IdentifyInteresting(solution)
    solution.pmi_score ← CalculatePMIScore(solution)

7: return FilterByPMIScore(creative_solutions, threshold=0.6)
```

3.4.4 RAG-Graph Knowledge Retrieval Algorithm

This algorithm manages the retrieval and integration of knowledge from both internal TRIZ databases and external sources through a graph-based RAG system.

```
Algorithm: RAGGraphRetrieval
Input: query, knowledge_graph, external_sources
Output: relevant_knowledge, confidence_scores

1: // Query Embedding and Expansion
```

```
  query_embedding ← GenerateEmbedding(query)
  expanded_queries ← QueryExpansion(query, synonyms=true, related_terms=true)

2: // Graph Traversal
  initial_nodes ← FindNearestNodes(query_embedding, knowledge_graph, k=10)
  relevant_nodes ← []

3: for each node in initial_nodes:
    // Breadth-first search with semantic similarity
    neighbors ← BFS(node, max_depth=3)
    for each neighbor in neighbors:
      similarity ← CosineSimilarity(query_embedding, neighbor.embedding)
      if similarity > SIMILARITY_THRESHOLD:
        relevant_nodes.append({
          'node': neighbor,
          'similarity': similarity,
          'path': GetPath(node, neighbor)
        })

4: // External Source Integration
  external_results ← []
  for each source in external_sources:
    if source.type == 'web_search':
      results ← GoogleSearch(expanded_queries, max_results=5)
    elif source.type == 'wikipedia':
      results ← WikipediaAPI(expanded_queries)
    elif source.type == 'patent_db':
      results ← PatentSearch(expanded_queries)

    external_results.extend(results)

5: // Knowledge Fusion
  combined_knowledge ← FuseKnowledge(relevant_nodes, external_results)

6: // Confidence Scoring
  for each item in combined_knowledge:
    item.confidence ← CalculateConfidence(
      source_reliability=item.source.reliability,
      semantic_similarity=item.similarity,
      corroboration=CountCorroborating(item, combined_knowledge)
    )

7: // Relationship Extraction
  relationships ← ExtractRelationships(combined_knowledge)
  knowledge_subgraph ← ConstructSubgraph(combined_knowledge, relationships)
```

```
8: return knowledge_subgraph, SortByConfidence(combined_knowledge)
```

3.4.5 Evolutionary Solution Refinement Algorithm

Inspired by AlphaEvolve, this algorithm implements an evolutionary approach to iteratively refine and improve solution candidates through mutation and selection.

Algorithm: EvolutionaryRefinement

```
Input: initial_solutions, fitness_criteria, max_generations
Output: evolved_solutions

1: population ← initial_solutions
   generation ← 0

2: while generation < max_generations:
      // Fitness Evaluation
      for each solution in population:
         solution.fitness ← EvaluateFitness(solution, fitness_criteria)

      // Selection
      parents ← TournamentSelection(population, tournament_size=3)

      // Crossover
      offspring ← []
      for i in range(0, len(parents), 2):
         if Random() < CROSSOVER_RATE:
            child1, child2 ← CrossoverSolutions(parents[i], parents[i+1])
            offspring.extend([child1, child2])
         else:
            offspring.extend([parents[i], parents[i+1]])

      // Mutation
      for each solution in offspring:
```

```
      if Random() < MUTATION_RATE:
        solution ← MutateSolution(solution)

    // LLM-Enhanced Mutation
    creative_mutations ← []
    for each solution in SelectBest(offspring, n=5):
      mutated ← LLMGenerate(
        prompt="Creatively modify this solution: {solution}",
        temperature=0.8  // Higher temperature for creativity
      )
      creative_mutations.append(mutated)

    offspring.extend(creative_mutations)

    // Environmental Selection
    population ← ElitistSelection(population + offspring, size=POPULATION_SIZE)

    // Convergence Check
    if CheckConvergence(population):
      break

    generation += 1

3: evolved_solutions ← SelectBest(population, n=10)
4: return evolved_solutions

Function: EvaluateFitness(solution, criteria)
  fitness ← 0
  fitness += criteria.novelty_weight * AssessNovelty(solution)
  fitness += criteria.feasibility_weight * AssessFeasibility(solution)
  fitness += criteria.ideality_weight * AssessIdeality(solution)
  fitness += criteria.resource_weight * AssessResourceEfficiency(solution)
  return fitness
```

3.4.6 Intuition Quality Assessment Algorithm

This algorithm quantifies the "intuitive quality" of generated solutions by analyzing patterns similar to expert human judgment.

Algorithm: IntuitionQualityAssessment

Input: solution, expert_patterns, context
Output: intuition_score, quality_metrics

```
1: // Pattern Recognition Score
   pattern_matches ← []
   for each pattern in expert_patterns:
      match_strength ← CalculatePatternMatch(solution, pattern)
      if match_strength > PATTERN_THRESHOLD:
         pattern_matches.append({
            'pattern': pattern,
            'strength': match_strength
         })

2: pattern_score ← AggregatePatternScores(pattern_matches)

3: // Non-Obviousness Metric
   baseline_solutions ← GenerateBaselineSolutions(context)
   semantic_distance ← AverageSemanticDistance(solution, baseline_solutions)
   non_obviousness ← Normalize(semantic_distance, 0, 1)

4: // Elegance Assessment
   complexity ← MeasureComplexity(solution)
   effectiveness ← MeasureEffectiveness(solution, context)
   elegance ← effectiveness / (1 + complexity)  // Simplicity-effectiveness ratio

5: // Cross-Domain Applicability
   domains ← IdentifyApplicableDomains(solution)
   generalizability ← len(domains) / MAX_DOMAINS

6: // Cognitive Load Analysis
   cognitive_load ← EstimateCognitiveLoad(solution)
   accessibility ← 1 - Normalize(cognitive_load, 0, 1)

7: // System 1 vs System 2 Balance
   analytical_components ← ExtractAnalyticalComponents(solution)
   intuitive_components ← ExtractIntuitiveComponents(solution)
   cognitive_balance ← CalculateBalance(analytical_components, intuitive_components)

8: // Aggregate Intuition Score
   intuition_score ← WeightedAverage([
      (pattern_score, 0.25),
```

```
    (non_obviousness, 0.20),
    (elegance, 0.20),
    (generalizability, 0.15),
    (accessibility, 0.10),
    (cognitive_balance, 0.10)
  ])

9: quality_metrics ← {
    'pattern_recognition': pattern_score,
    'non_obviousness': non_obviousness,
    'elegance': elegance,
    'generalizability': generalizability,
    'accessibility': accessibility,
    'cognitive_balance': cognitive_balance,
    'overall_intuition': intuition_score
  }

10: return intuition_score, quality_metrics
```

3.4.7 Multi-Agent Coordination Algorithm

This algorithm manages the coordination between multiple specialized agents using LangGraph for internal orchestration and n8n for external system integration.

Algorithm: MultiAgentCoordination

```
Input: task, agent_pool, coordination_strategy
Output: coordinated_result, execution_trace

1: // Agent Initialization
  agents ← {
    'analyzer': InitializeAnalyzerAgent(),
    'creative': InitializeCreativeAgent(),
    'critic': InitializeCriticAgent(),
    'synthesizer': InitializeSynthesizerAgent()
  }

2: // Task Decomposition
  subtasks ← DecomposeTask(task)
```

```
  task_graph ← BuildTaskDependencyGraph(subtasks)

3: // LangGraph Internal Orchestration
  internal_state ← InitializeLangGraphState()

4: for each subtask in TopologicalSort(task_graph):
    assigned_agent ← SelectAgent(subtask, agents)

    // Agent Execution with Memory
    agent_input ← PrepareAgentInput(subtask, internal_state)
    agent_output ← assigned_agent.Execute(agent_input)

    // State Update
    internal_state ← UpdateState(internal_state, agent_output)

    // Checkpoint for Human-in-the-Loop
    if RequiresHumanValidation(subtask):
      human_feedback ← RequestHumanFeedback(agent_output)
      agent_output ← IncorporateFeedback(agent_output, human_feedback)

5: // n8n External Orchestration
  external_workflow ← {
    'nodes': [],
    'connections': []
  }

6: for each external_task in IdentifyExternalTasks(task):
    node ← CreateN8NNode(external_task)
    external_workflow.nodes.append(node)

    // Define connections based on dependencies
    dependencies ← GetDependencies(external_task)
    for each dep in dependencies:
      connection ← CreateConnection(dep, node)
      external_workflow.connections.append(connection)

7: // Execute External Workflow
  external_results ← ExecuteN8NWorkflow(external_workflow)

8: // Merge Internal and External Results
  coordinated_result ← MergeResults(internal_state, external_results)

9: // Generate Execution Trace for Transparency
  execution_trace ← GenerateTrace(internal_state, external_workflow)
```

```
10: return coordinated_result, execution_trace
```

4. Implementation and Architecture

4.1 Frontend

The interface uses React and Electron to provide a desktop environment with keyboard navigation, content bookmarking, knowledge graph interaction, and paginated document views.

4.2 Backend

- **LLM:** GPT-4 / Gemini 2.5 Pro
- **Knowledge Graph:** Neo4j
- **RAG-Graph:** Integrates internal TRIZ knowledge base with external sources
- **Microservices Architecture with an API Gateway :** Python (FastAPI) + zerorpc
- **Orchestration Flow Processing:** n8n + LangGraph for orchestration

4.3 Workflow Design

Data is collected from sources including the web, Wikipedia, and curated knowledge repositories. RAG-Graph constructs a knowledge graph, which is then processed by the TRIZ and lateral thinking modules. LangGraph+n8n manages and displays the process chain for transparency.

5. Framework Validation Approach and Challenges

5.1 Framework Validation Strategy

- Accuracy and clarity of generated content
- Novelty, interpretability, and feasibility of solution
- Knowledge graph consistency
- User perception of “intuitive” quality in outputs
- Diversity of proposed solutions

5.2 Anticipated Challenges

- Quantifying “intuition” in measurable terms[8]
- Mitigating hallucinations from LLMs
- Real-time orchestration of multi-agent workflows
- Ethical considerations in AI-generated decision-making
- Computational cost of complex agent coordination

6. Conclusion and Future Work

The IIA framework advances AI problem-solving by combining TRIZ's structured methodology with the creative flexibility of lateral thinking, operationalized through LLMs, RAG, and n8n. This approach addresses both the logical and intuitive dimensions of inventive reasoning.

Contributions:

- Unified automation of TRIZ and lateral thinking within a single framework
- Formal modeling of intuition using System 1 and System 2 principles
- Integration of LLMs, multi-agent coordination, and human oversight

Future Directions:

- Develop formal metrics for evaluating intuitive correctness
- Extend framework to multimodal reasoning (text, diagrams, images)
- Implement adaptive learning through RLHF and expert feedback
- Conduct comparative studies with expert human innovators
- Deeper study and incorporation of AlphaEvolve principles.

By operationalizing both structured logic and intuitive reasoning, this work takes a step toward AI systems that can think, adapt, and innovate with the skill and creativity of a human inventor.

7. References

[1] K. Christakopoulou, S. Mourad, and M. Mataríć, “Agents thinking fast and slow: A talker-reasoner architecture,” *arXiv preprint* arXiv:2410.08328, Oct. 2024, doi: 10.48550/arXiv.2410.08328.

[2] S. Jiang and J. Luo, “AutoTRIZ: Artificial ideation with TRIZ and large language models,” *arXiv preprint* arXiv:2403.13002, Mar. 2024

[3] L. Chen, Y. Song, S. Ding, L. Sun, P. Childs, and H. Zuo, “TRIZ-GPT: An LLM-augmented method for problem-solving,” *arXiv preprint* arXiv:2408.05897, Aug. 2024, doi: 10.48550/arXiv.2408.05897.

[4] K. Szczepanik and J. A. Chudziak, “TRIZ agents: A multi-agent LLM approach for TRIZ-based innovation,” *arXiv preprint* arXiv:2506.18783, Jun. 2025, doi: 10.48550/arXiv.2506.18783.

[5] Q. Chen, B. Zhang, G. Wang, and Q. Wu, “Weak-eval-Strong: Evaluating and eliciting lateral thinking of LLMs with situation puzzles,” *arXiv preprint* arXiv:2410.06733, Oct. 2024, doi: 10.48550/arXiv.2410.06733.

[6] Z. Huang *et al.*, “Reinforcing the diffusion chain of lateral thought with diffusion language models,” *arXiv preprint* arXiv:2505.10446, May 2025, doi: 10.48550/arXiv.2505.10446.

[7] S. Dernbach *et al.*, “Thinking fast and laterally: Multi-agentic approach for reasoning about uncertain emerging events,” *arXiv preprint* arXiv:2412.07977, Dec. 2024, doi: 10.48550/arXiv.2412.07977.

[8] K. Zhang and H. Lipson, “Aligning AI-driven discovery with human intuition,” *arXiv preprint* arXiv:2410.07397, Oct. 2024, doi: 10.48550/arXiv.2410.07397.

[9] A. Novikov et al., “AlphaEvolve: A coding agent for scientific and algorithmic discovery,” arXiv preprint arXiv:2506.13131, Jun. 2025, doi: 10.48550/arXiv.2506.13131.