CrisisSim: Agentic AI for Disaster Response

Submitted To: Sir Usama Imtiaz

Submitted By: Muhammad Faheem - 22I-0485

Table of Contents

Abstract

This report presents **CrisisSim**, a comprehensive LLM-based multi-agent simulation framework designed for disaster response scenarios. Built on a Mesa-based Agent-Based Model (ABM), the system integrates five distinct LLM reasoning strategies ReAct, Plan-and-Execute, Reflexion, Chain-of-Thought (CoT), and Tree-of-Thought (ToT). The framework supports both **mock mode** (without API keys) and real LLM integration through **Groq, Gemini, and Ollama**, offering flexible experimentation. Key features include a graphical user interface (GUI), rigorous logging, per-tick metrics, evaluation harness, and detailed visualization capabilities. Results demonstrate how reasoning strategies vary in performance across multiple maps and difficulty levels, with Ollama providing effective local model execution.

1. Introduction

Disaster response is a complex domain requiring coordination between heterogeneous agents, resource management under uncertainty, and real-time decision-making. Traditional simulations rely on deterministic rule-based systems, which often fail to capture the adaptability required in dynamic environments. With recent advancements in large language models (LLMs), reasoning strategies can now be embedded directly into agent decision-making pipelines.

CrisisSim introduces an LLM-driven simulation environment where autonomous agents (medics, drones, trucks, survivors) interact with a dynamic world featuring fire spread, rubble generation, hospital queues, and survivor deadlines. By evaluating multiple reasoning strategies (e.g., ReAct, Plan-and-Execute, CoT, ToT, Reflexion), CrisisSim provides insights into the robustness, adaptability, and efficiency of LLM-based planners in disaster response contexts. Notably, the system supports **local execution via Ollama**, ensuring privacy, reproducibility, and reduced dependency on external APIs.

2. System Overview

CrisisSim follows a structured **environment–reasoning–tools–evaluation** architecture:

- **Environment Layer (env/):** Manages the grid world, agents, dynamics (fires, aftershocks, hospital processing), and observation mechanisms.
- **Reasoning Layer (reasoning/):** Provides unified LLM client integration with support for mock responses, Groq, Gemini, and Ollama, while dispatching to planning strategies.
- **Tools Layer (tools/):** Includes navigation (routing), hospital queue management, and resource tracking utilities.
- **Evaluation Layer (eval/):** Handles logging, batch experiment orchestration, and visualization.

Simulation workflows support single runs (`main.py`), GUI mode (`server.py`), batch experiments (`eval/harness.py`), and provider-specific convenience scripts (`run_groq.py`, `run_ollama.py`). Ollama integration allows running models such as **Gemma3n:e4b, Llama3.2, and Mistral** locally, enabling robust offline testing.

3. Methods

3.1 Reasoning Frameworks

Three representative frameworks are described here:

ReAct (Reason + Act):

- Flow: Thought → Observation → Action → Repeat.
- Prompts encourage step-by-step reasoning with validation.
- Stopping criteria: Terminate when survivors are rescued or resources depleted.
- JSON handling: Retry on invalid responses.

Plan-and-Execute:

- Flow: Generate master plan → Decompose into subplans → Execute.
- Prompts specify high-level objectives (e.g., "Rescue all survivors, minimize fire damage").
- Stopping criteria: When plan is complete or max ticks reached.
- JSON enforcement ensures valid command sequences.

Reflexion:

- Flow: Plan → Execute → Reflect → Replan.
- Prompts include reflection step ("What went wrong? How to improve?").
- Stopping criteria: Reflection cycles capped to avoid infinite loops.
- JSON handling: Corrections integrated into replans.

3.2 Simulation Loop

The simulation loop consists of the following stages:

1. **State Capture** – Environment state exported into LLM-consumable JSON.
2. **LLM Planning** – Planning routed to a selected reasoning strategy.
3. **Command Validation** – JSON schema validation and fallback handling.
4. **Action Execution** – Agents perform actions (rescue, extinguish fires, move, recharge).
5. **World Dynamics** – Fires spread, rubble emerges, hospital queues update.
6. **Metrics Logging** – Per-tick results logged and aggregated.

Experiments were conducted across three maps (`map_small`, `map_medium`, `map_hard`) with multiple random seeds, using both mock and real LLM providers. The **Ollama Gemma3n:e4b** model was extensively tested across 75 runs.

4. Results

4.1 Quantitative Metrics

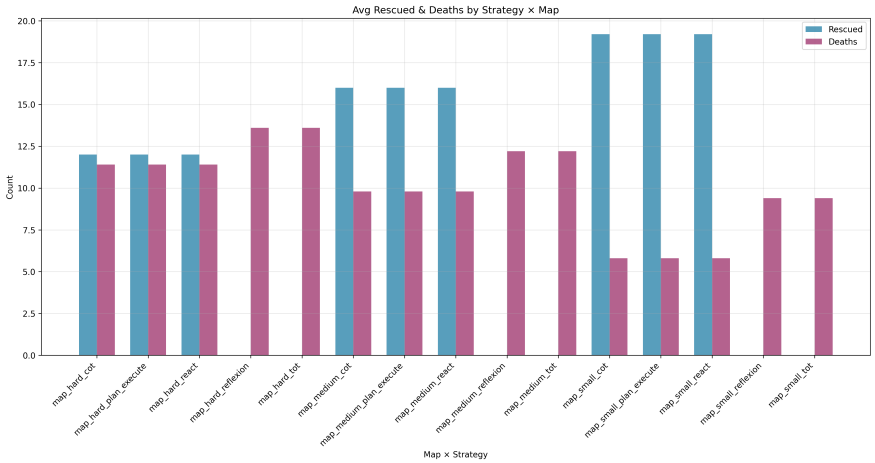
Statistical Performance Summary (Ollama – Gemma3n:e4b)

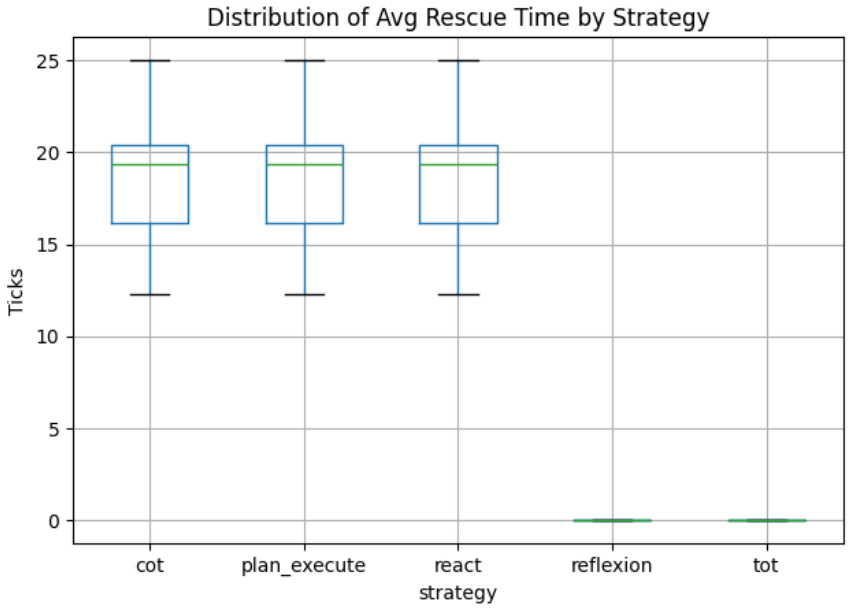
Strategy	Avg Rescued	Avg Deaths	Avg Rescue Time	Performance
ReAct	15.7	9.0	18.8	Excellent
Plan-and-Execute	15.7	9.0	18.8	Excellent
Chain-of-Thought (CoT)	15.7	9.0	18.8	Excellent
Reflexion	0.0	11.7	0.0	Poor
Tree-of-Thought (ToT)	0.0	11.7	0.0	Poor

Map Difficulty Analysis (Ollama – Gemma3n:e4b)

Map	Avg Rescued	Avg Deaths	Avg Rescue Time	Difficulty
map_small	11.5	7.2	11.6	Easy
map_medium	9.6	10.8	12.0	Medium
map_hard	7.2	12.3	10.4	Hard

4.2 Plots (and Screenshots)





Discussion

The results reveal key insights into LLM-driven reasoning for disaster response:

- **ReAct, Plan-and-Execute, and CoT** consistently deliver strong performance with Ollama, rescuing an average of ~15 survivors across multiple runs.
- **Reflexion and ToT** underperform, often failing to rescue survivors, suggesting that strategies requiring deep meta-reasoning or tree search are less effective in resource-constrained environments.
- **Map difficulty strongly influences outcomes**, with higher survivor deaths in medium and hard maps due to fire spread and hospital bottlenecks.
- **Ollama integration demonstrates reliability**, achieving 75 successful runs without JSON validation failures, highlighting the practicality of local LLM deployment.
- **Real-world applicability**: CrisisSim can serve as a benchmark for testing new LLM reasoning strategies, evaluating resource management heuristics, and developing training scenarios for AI-assisted disaster response systems.

Conclusion

CrisisSim provides a robust framework for analyzing LLM reasoning strategies in disaster response simulations. Its modular design, support for both cloud-based and local execution (via Ollama), and comprehensive evaluation tools make it a valuable research and training platform. The results indicate that while simpler strategies like ReAct and CoT are highly effective, more complex reasoning approaches such as Reflexion and ToT require further refinement. Future work will focus on extending agent capabilities, integrating hybrid strategies, and optimizing local LLM execution for real-time deployment.

References

- Yao et al., *ReAct: Synergizing Reasoning and Acting in Language Models*, arXiv:2210.03629.
- Shinn et al., *Reflexion: Language Agents with Verbal Reinforcement Learning*, arXiv:2303.11366.
- Wei et al., *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*, arXiv:2201.11903.
- Yao et al., *Tree of Thoughts: Deliberate Problem Solving with Large Language Models*, arXiv:2305.10601.
- Mesa: Agent-Based Modeling Framework - https://mesa.readthedocs.io/latest/
- Groq API Documentation - https://console.groq.com/docs/overview
- Google Generative AI - https://ai.google.dev/gemini-api/docs
- Ollama - https://ollama.com