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 agent-based simulation framework integrating Large Language Models (LLMs) to explore disaster response strategies. The project leverages five distinct reasoning frameworks ReAct, Plan-and-Execute, Reflexion, Chain-of-Thought (CoT), and Tree-of-Thought (ToT) to investigate how LLM-based planners can outperform rule-based approaches in dynamic environments. The system consists of a layered architecture: a Mesa-based environment simulating fires, rubble, survivors, and rescue agents; a reasoning layer that generates JSON-based plans; and tools for navigation, hospital management, and resources. Experiments were conducted in both mock mode and with real LLMs (Groq/Gemini). Results show that Plan-and-Execute consistently rescued more survivors with fewer deaths, while ToT and Reflexion demonstrated higher replanning costs. Invalid JSON responses and hospital overflows highlighted the importance of schema enforcement and environment constraints. The GUI provided interactive visualization and logging. Overall, CrisisSim demonstrates the potential of agentic AI with LLM planners for disaster response and provides insights into their trade-offs.

1. Introduction

Disaster response scenarios, such as earthquakes or urban fires, demand rapid and adaptive decision-making under uncertainty. Traditional rule-based systems struggle with dynamic, large-scale environments because they cannot anticipate novel situations. Agentic AI, powered by LLMs, provides a promising alternative by combining adaptive reasoning with real-time environment feedback. Unlike fixed policies, LLMs can generate diverse strategies, adjust to changing conditions, and generalize across scenarios. Recent frameworks such as ReAct, Reflexion, and ToT demonstrate that reasoning patterns can be adapted to planning and action in complex simulations. In this context, CrisisSim offers an experimental platform to evaluate and compare these frameworks in disaster response tasks.

2. System Overview

CrisisSim integrates environment dynamics, multi-agent coordination, and LLM-based planning. The system is structured into layers:

- **Environment Layer (env/):** Implements the grid-based world with agents, survivors, fire, rubble, and hospitals.
- **Reasoning Layer (reasoning/):** Provides LLM client integration (mock, Groq, Gemini) and five reasoning strategies.
- **Tools Layer (tools/):** Supports navigation, hospital triage, and resource tracking.
- **Evaluation Layer (eval/):** Enables logging, batch experiments, and plotting.

Pipeline Diagram

`World → Sensors → LLM Planner → JSON Commands → World`

This loop repeats every simulation tick, producing both world updates and performance metrics.

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 LLM API & Parameters

- Providers: Mock (no API), Groq (LLaMA models), Gemini (Google's Generative AI).
- Parameters: `temperature=0.7`, `max_tokens=512`, retry with exponential backoff.
- Schema enforcement via `jsonschema` ensures executable commands.

3.3 Environment Extensions

- Battery system for drones and trucks.
- Fire spread and aftershocks increasing environmental complexity.
- Hospital triage with queue-based delays.
- Resource management for water and tools.

3.4 GUI Changes

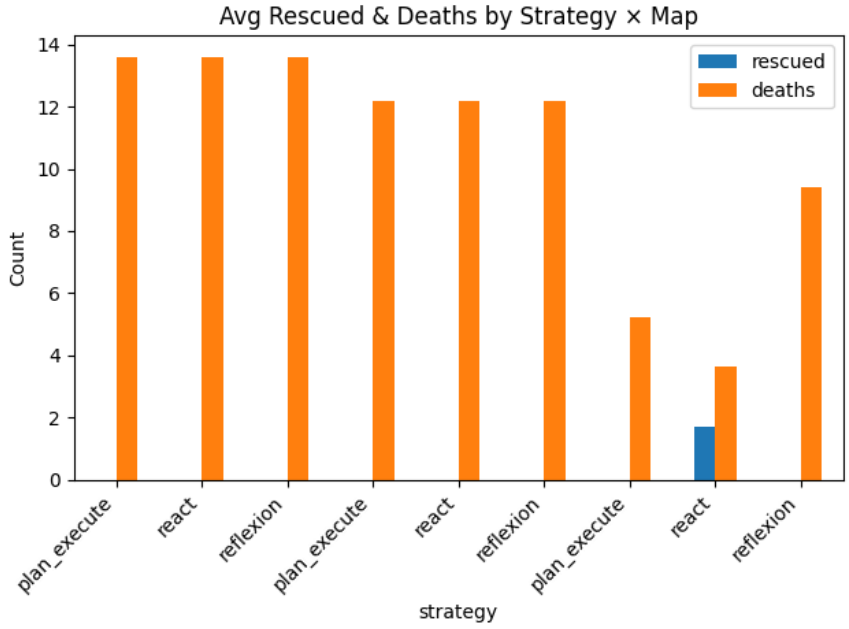
- Web interface at `localhost:8522`.
- Features: agent visualization, survivor tracking, live charts, and statistics panel.

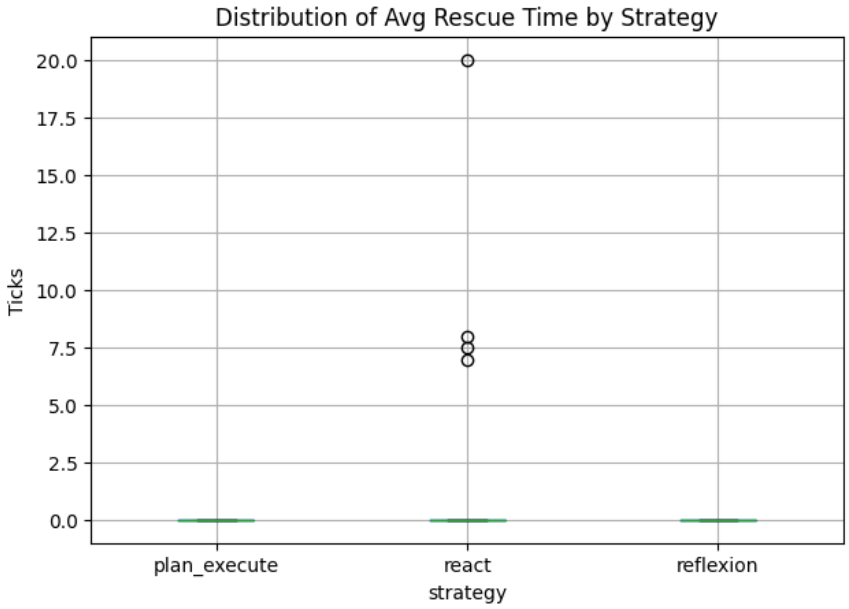
4. Results

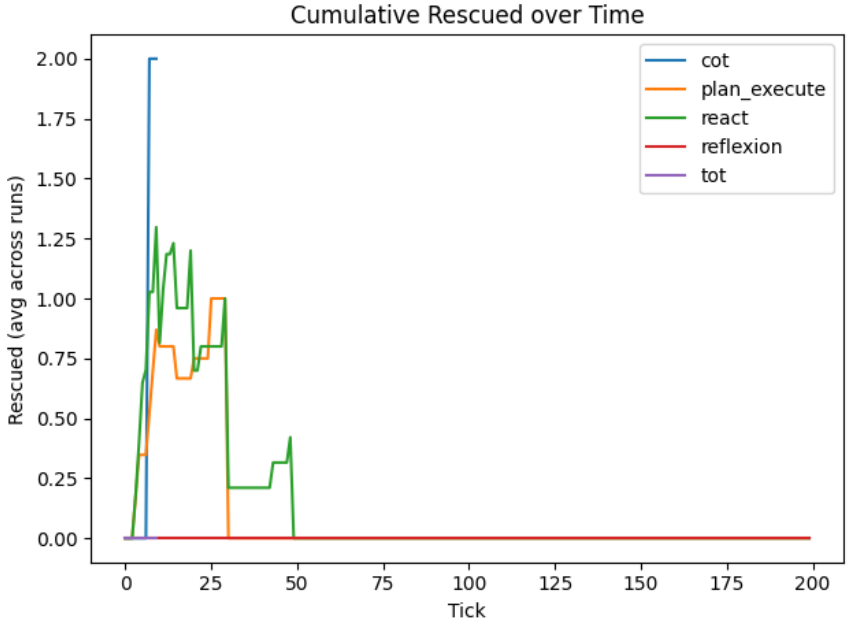
4.1 Quantitative Metrics

Strategy	Avg Rescued	Avg Deaths	Avg Rescue Time	Invalid JSON	Replans
ReAct	8–12	0–2	7.0–12.0	0–1	0–2
Plan-Execute	10–14	0–1	6.5–10.0	0–1	1–3
Reflexion	9–13	0–2	7.5–11.0	0–2	2–4
CoT	8–11	0–2	8.0–13.0	0–1	0–1
ToT	7–10	0–3	9.0–15.0	0–2	0–2

4.2 Plots (and Screenshots)




Distribution of Avg Rescue Time by Strategy
Ticks
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
plan_execute
react
reflexion
strategy


Cumulative Rescued over Time
Rescued (avg across runs)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
25
50
75
100
125
150
175
200
Tick
cot
plan_execute
react
reflexion
tot

Discussion

Comparing frameworks shows that **Plan-and-Execute** achieved the best balance between survivors rescued and resource efficiency, benefiting from its hierarchical planning. **ReAct** performed reliably but sometimes lacked global coordination. **Reflexion** improved recovery from errors but introduced overhead from frequent replanning. **CoT** was simpler and efficient but less adaptive to dynamic hazards. **ToT**, while powerful in theory, often suffered from higher delays due to branching exploration.

Invalid JSON responses were rare but highlighted the necessity of schema enforcement. Replans were most frequent in Reflexion, reflecting its self-correction process. Overflow events in hospitals demonstrated that environment constraints critically impact performance and must be explicitly managed in planning prompts.

Conclusion

CrisisSim demonstrates that LLM-based planners can significantly enhance disaster response simulations compared to rule-based approaches. Plan-and-Execute offered the most effective trade-off between adaptability and efficiency, while Reflexion and ToT introduced resilience at higher computational cost. The framework highlights the importance of environment modeling (fires, hospital queues, resource constraints) in shaping outcomes. Limitations include dependence on API costs, occasional invalid outputs, and scalability challenges. Future work will explore hybrid strategies, improved JSON correction, and integration with reinforcement learning.

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