Project Details for CrewForge: Agents that Hire Agents for Adaptive Teaming

Amirhossein Sabour, Mehdi Moradi

McMaster University, Hamilton, ON, Canada
saboua4@mcmaster.ca

Abstract. Most existing multi-agent LLM systems rely on fixed agent roles and static communication patterns tailored to specific tasks, limiting their adaptability and generality. We introduce CrewForge, a task-agnostic framework where a Meta Agent dynamically designs, prunes, and rewires teams of communicative agents based on the task description and ongoing performance. The framework can support mid-task reflection and adaptation, enabling agents to be spawned, retired, or reconfigured during execution. Evaluation can be done in a task-specific manner across a range of Python-accessible environments, including strategic, cooperative, and social games. CrewForge's goal is to reduce manual role engineering which can be task-specific and sub-optimal, and facilitate the development of general-purpose, self-organizing multi-agent LLM systems.

1 Motivation

Most multi-agent communicative LLM systems fix roles and wiring ahead of time and test in one setting. Agents of Change [1] is the first to show self-evolving capabilities of LLM agents in Settlers of Catan yet the pipeline is game-specific and evaluated on a single title. We aim for a superset of this approach that generalizes across games with roles created and pruned dynamically by a meta agent. [1] shows self improvement in a strategic board game setting using a Catan simulator. It is compelling yet game bound. Our proposed framework keeps the self evolution idea removes the need to concretely define task-specific agents and their respective roles, which is considered a hindrance in general task-agnostic application of communicative multiagent LLM systems.

2 Proposed System

2.1 Meta Agent

Input environment card, rules, action space, reward spec, team size budget, communication budget. **Output** a small set of role specs with persona, responsibilities, tools, and message schema. **Mid game** observe traces and metrics then merge roles, retire agents, or spawn a new role.

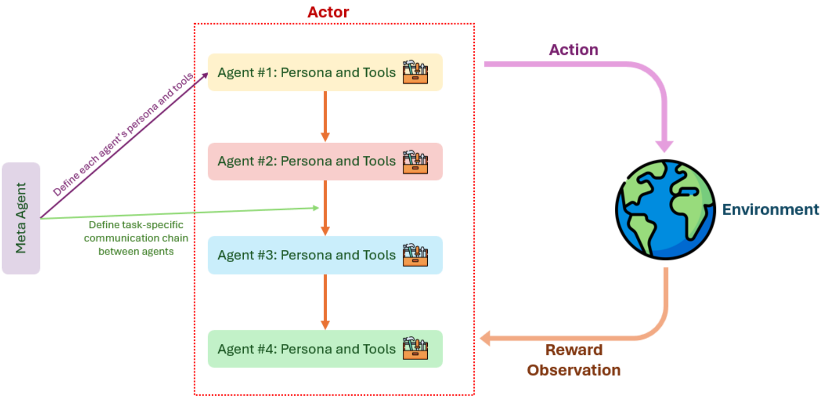
Actor
Agent #1: Persona and Tools
Agent #2: Persona and Tools
Agent #3: Persona and Tools
Agent #4: Persona and Tools
Meta Agent
Define each agent's persona and tools
Define task-specific communication chain between agents
Action
Environment
Reward
Observation

Fig. 1: Overview of the CrewForge architecture. Given a comprehensive task description, the Meta Agent defines the roles, personas, tools, and communication chain for the task-specific agents. These agents are then instantiated and interact with the environment. Mid-task, the Meta Agent can reflect and decide whether to retain or reconfigure the current agent setup and communication chain, making the framework adaptive and task-agnostic.

2.2 Role Schema

Each role has a prompt template, a validated JSON interface for actions and messages, a tool list (rulebook lookup, short planner, opponent notes), and a memory layout with strict size caps.

2.3 Control Loop

Plan then Play then Reflect then Adapt. The meta agent proposes a team and routing. Agents act within token budgets. Short self critique and peer review. If triggers fire the meta agent edits personas or the roster of agent based on its reflect and the currect state of the task/game.

2.4 Adaptation Triggers

The triggers that can potentially be used to by the Meta Agent include but are not limited to: Rolling performance drop, high conflict rate in messages, repeated tool failures, or opportunity signals from the world model. Changes are rate limited and can pass a simple cost benefit check.

3 Target Environments

The framework is task-agnostic and can be applied to a wide range of target environments. In practice, gaming environments that involve cooperation, strategic thinking, and planning toward a common goal are ideal candidates. One limitation of the framework is its reliance on multiple forward passes and inter-agent communication, which may reduce its effectiveness in time-sensitive games that require quick reactions. Environments, tasks, and potential games will be selected based on their availability in Python and their adaptability for incorporating multi-agent LLM systems. Examples include but are not limited to strategic trading games such as Settlers of Catan, time-pressured coordination games such as Overcooked, grid-based adversarial games, social deduction games such as Mafia or Werewolf, and even MMORPGs, depending on whether they offer Python APIs or support cost-effective integration of communicative agents.

4 Research Questions

- Does meta designed composition outperform fixed teams under equal budgets?
- Does mid game revision improve returns compared with revision between games only?
- Which role primitives transfer across games without reprompting?

5 Metrics

Win rate or score. Steps to reach a fixed score. Token cost per successful action. Adaptation value measured as the change in performance within N steps after a roster change. Stability across seeds and partners.

6 Why This Is a Good Fit

This pushes dynamic adaptability into a research area that is mostly focused on using static methods to develop task-specific agents today. It reduces per environment engineering and turns role design into a learned and auditable component.

References

1. Belle, N., Barnes, D., Amayuelas, A., Bercovich, I., Wang, X.E., Wang, W.: Agents of change: Self-evolving llm agents for strategic planning. arXiv preprint arXiv:2506.04651 (2025)