LITERATURE REVIEW

Multi-Agent Systems for Business Trip Planning

Using Local Large Language Models

Research Question:

How can a multi-agent system run on local LLMs improve efficiency in planning business trips?

Laureys Indy

Bachelor Creative Technologies & AI

Howest University of Applied Sciences

Innovation & Research Project

Table of Contents

1. Introduction

This literature review looks at the theoretical foundations and current state of research that is relevant to developing a multi-agent system for automated business trip planning using locally deployed large language models (LLMs). The research addresses the central question: *How can a multi-agent system run on local LLMs improve efficiency in planning business trips?*

Business travel coordination is a complex planning problem involving multiple constraints: flight schedules, hotel availability, budget limitations... Traditional approaches require you to manually visit multiple booking websites, which consumes time but also might overlook certain constraints (such as budget and timing). Multi-agent systems offer a solution where specialized agents can work together to break down this complexity into manageable smaller tasks while maintaining structured communication.

1.1 Research Sub-Questions

The theoretical sub-questions guiding this review are:

- What impact does agent specialization have on planning success rates and iteration counts?
- How do feedback loops between verification and booking agents vary with policy strictness?
- How does trip complexity affect message exchanges and end-to-end planning time?
- How does prompt specificity influence policy violation rates and booking accuracy?

1.2 Scope and Methodology

This review used peer-reviewed publications from major platforms including ICML, ICLR, ACL, IJCAI, and NeurIPS, alongside the technical documentations from industry leaders (Anthropic, Microsoft, LangChain). Sources were selected based on the relevance to multi-agents, LLM-based agents and travel planning systems. Publications from 2023-2025 were prioritized as they give the most up-to-date findings on this rapidly evolving field of AI agents.

2. Multi-Agent System Fundamentals

2.1 Theoretical Foundations

Over the past three decades, multi-agent systems (MAS) have transformed from largely theoretical concepts to practical, real-world applications. Among the models proposed for intelligent agents, the **Belief-Desire-Intention (BDI) architecture**, introduced by Rao and Georgeff [1], has become the most impactful. In this framework, agents reason about their environment using beliefs, pursue objectives represented as desires, and commit to specific sources of action through intentions. Agent behaviour evolves through a continuous cycle of processing events, updating beliefs, selecting appropriate plans, and executing actions.

2.2 Coordination Models

Coordination models play an important role in shaping how a system behaves. A common approach is the **hub-and-spoke (centralized) model**, where a single orchestrator directs a set of specialized worker agents. This setup offers good control and consistent execution, making it ideal for structured business workflows. Tran et al. [2] point out in their 2025 survey that this “puppeteer-style” coordination can organize agent activities on the fly, boosting performance while minimizing computational costs.

The Contract Net Protocol (CNP), introduced by Reid Smith [3], is the main method for assigning tasks in multi-agent systems. It works by defining manager and contractor roles. It also follows a clear process: call for proposals → bids → contract award → execution. For example, in trip planning, the orchestrator might send a request like “find flights under €500 that arrive in the morning” and the specialized agents respond to this request with bids based on what they can do but also how busy they are.

2.3 Communication Patterns

For multi-agent systems to work well together, they need clear communication protocols. The **FIPA Agent Communication Language (ACL)** [4] sets a standard way for agents to structure their messages. FIPA-ACL defines fields like performative, sender, receiver, content, ontology, and conversation-id. Using this exact structure, agents can create custom message formats.

3. LLM-Powered Multi-Agent Systems

3.1 Architecture and Components

The field of LLM-based multi-agent systems has grown rapidly since 2023. Guo et al.'s survey [5] has identified five essential agent components and how they work:

- profile (role specification)
- perception (environmental awareness)
- self-action (individual capabilities)
- mutual interaction (communication)
- evolution (learning and adaptation)

Chen et al. [6] extended this by categorizing systems by purpose: complex task solving, scenario simulation, and agent evaluation.

3.2 Framework Comparison

Several frameworks have been developed and created for building LLM-based multi-agent systems. Table 1 provides an analysis of the major options.

Table 1: Comparison of Multi-Agent Frameworks

Framework	Architecture	Strengths	Production Ready
LangGraph	Graph-based, event-driven	Full control, persistence, streaming, human-in-loop	High
AutoGen	Conversation-based	Flexible agents, no-code Studio, human input	Medium-High
CrewAI	Role-based crews	Simple API, native Ollama support	Medium
MetaGPT	SOP assembly line	Structured outputs, reduced hallucinations	Medium

LangGraph has become the best option for production-ready multi-agent systems. Developed by LangChain, it provides a low-level orchestration framework for stateful applications based on a graph architecture, where nodes are computation steps and edges define how control flows between them [7].

LangGraph provides important features, which includes durable execution (allowing systems to recover from failures), human-in-the-loop support (enabling inspection and modification of agent state at any stage) and built-in memory management. Its use in enterprise environments at companies like LinkedIn, Uber, and Klarna shows its suitability for real-world systems.

Microsoft's AutoGen [8] introduced customizable, interactive agents that combine large language models, human input and external tools using a conversation-based approach. In version 0.4 (released in January 2025), AutoGen moved to an asynchronous, event-driven design. While AutoGen is a valid option for flexible, multi-agent conversations, LangGraph has an advantage when precise control is needed. Because of its graph-based structure, it makes it easier to manage state transitions and supports workflows that cannot be done using simple conversational interactions.

4. Similar Studies in AI Travel Planning

4.1 The TravelPlanner Benchmark

The TravelPlanner Benchmark [9], presented by Xie et al., has been set as the golden standard for evaluating travel planning agents. It provides a controlled sandbox environment containing around 4 million data records and 1225 carefully designed planning tasks. The benchmark requires agents to plan transportation, meals, attractions, and accommodation while meeting multiple constraints like budget limits and time schedules.

The benchmark shows that GPT-4 achieves only a 0.6% success rate on complex planning tasks. This result emphasizes the limitations of standalone large language models, which often have difficulties with staying focused, choose the correct tools and tracking multiple constraints at once. These findings back up the use of multi-agent systems. The benchmark compares four planning approaches: Direct prompting, Zero-Shot, Chain-of-Thought, ReAct, and Reflexion across several models.

4.2 TravelAgent System

TravelAgent System [10] suggests a four-module architecture specifically designed for personalized travel planning. The system consists of a Tool-usage Module, Recommendation Module, Planning Module, and Memory Module. It combines specialized recommendation components for budget, city selection, flights, hotels, restaurants, and attractions with algorithms that take both time and location into account.

TravelAgent focuses on three main goals: rationality, completeness, and personalization. This method performs better than baseline GPT-4 based agents across all human evaluation studies. This design shows the benefits of separating issues within an agent system. These insights directly inform the proposed five-agent architecture for this project: Trip Orchestrator, Flight Agent, Hotel Agent, Policy Compliance Agent, and Time Management Agent.

4.3 Hybrid LLM-Solver Approaches

Hybrid approaches that combine both large language model reasoning with formal constraint-solving techniques can significantly improve performance. For example, TRIP-PAL [11], developed by J.P. Morgan AI Research, integrates LLMs with automated planning methods. In this system, the LLM converts travel information into structured data, while Mixed-Integer Programming is used to make sure that all constraints are met.

Another approach from the MIT-IBM Watson AI Lab sees travel planning as a verification problem. They developed a system where the LLM converts natural language into logic boundaries. The input is encoded into mathematical formulas. Using these encoded formulas, the solver can either find a solution or conclude that no solution exists at all with the given constraints.

This approach achieved a 97% success rate on the TravelPlanner benchmark, compared to just 0.6% for GPT-4 alone [12]. These results show that using verification methods alongside LLMs can turn reasoning into reliable constraint satisfaction. This idea is applied in the current project through the Policy Compliance Agent, which validates plans against a set of rules.

5. Technology Stack

5.1 Local LLM Deployment with Ollama

Ollama allows local deployment of LLMs by packaging model weights, configurations, and dependencies into portable containers [13]. It exposes a REST API on a port of your choice.

Ollama is built on llama.cpp [14], a lightweight C/C++ implementation of LLMs optimized for low-end hardware. It supports model quantization from 1.5-bit to 8-bit, making it possible to run efficient models on devices with limited resources.

Table 2: Local LLM Deployment Trade-offs

Factor	Local (Ollama)	Cloud API
Privacy	Data never leaves premises; GDPR compliant	Sensitive data transmitted externally
Latency	No network roundtrips	Network dependant; variable latency
Cost	One-time hardware investment	Recurring per-token charges
Scalability	Limited by hardware	Elastic scaling; high throughput
Model Access	Open-source models only	Access to frontier models

For proof-of-concept development, Ollama is a good choice because its benefits in privacy, low cost, and independence from external APIs outweigh its scalability limitations. Models such as Llama 3.2 and Mistral have enough reasoning capability for travel planning tasks.

5.2 Agent Orchestration with LangGraph

LangGraph is a library built on top of LangChain that is designed for creating stateful, multi-agent applications using LLMs [8]. Instead of relying on simple step-by-step chains, LangGraph uses a directed graph structure where nodes represent agents and edges control how information flows between them.

The framework uses a state-based architecture where all agents share a common typed state dictionary. Updates to this state are handles atomically and persist across node executions, which helps prevent race conditions that often occur. Each node in the graph can read the current state, perform computations and return partial updates that are automatically merged back into the shared state.

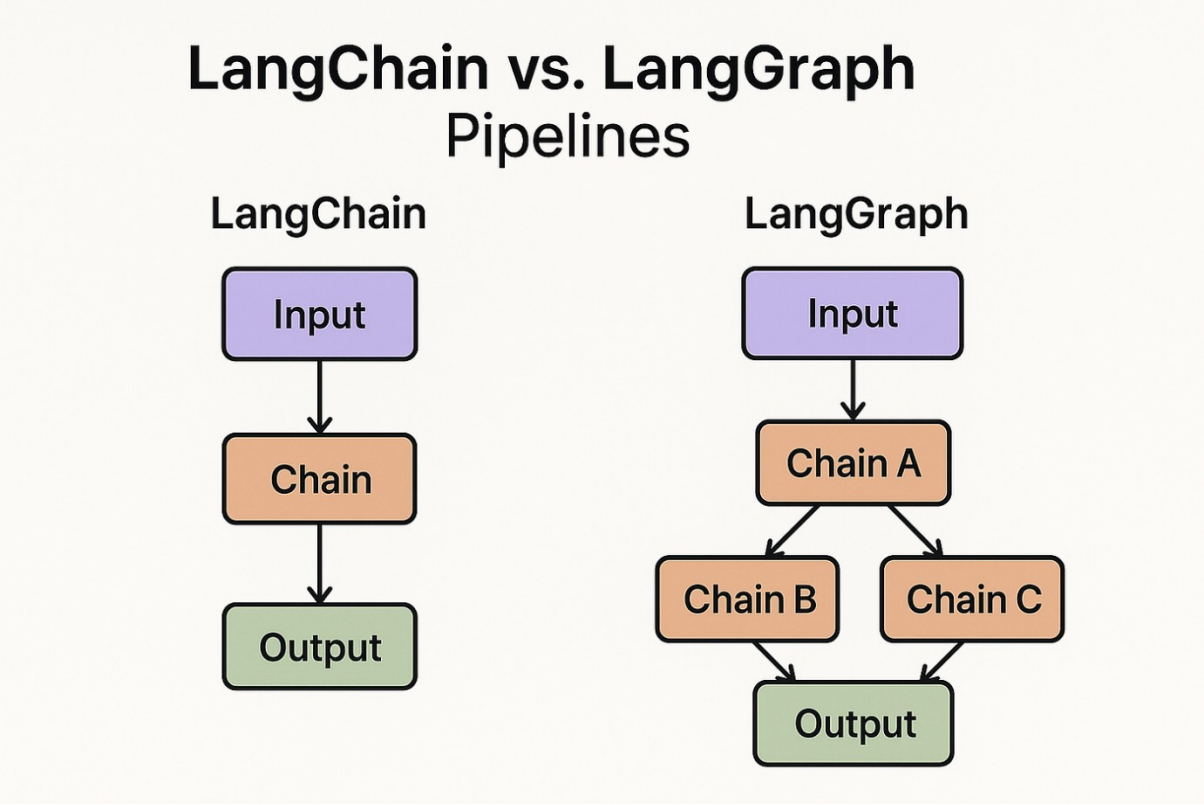

Figure 1: LangGraph workflow visualization [8]

5.3 Backend Framework with FastAPI

FastAPI is a Python web framework for building APIs with built-in support for data validation, serialization, and automatic documentation generation. It is built on Starlette for web handling and Pydantic for data validation. FastAPI also supports async/await, which is important for handling multiple LLM requests from different agents. The framework async capabilities are essential. FastAPI can handle multiple tasks at once. This eliminates the performance drop of sequential API calls to Ollama.

FastAPI automatically generates OpenAPI (Swagger) documentation, allowing frontend developers to understand APIs without manual documentation. Request validation through Pydantic models ensures that wrong inputs are rejected before reaching the LangGraph Orchestrator.

6. Prompt Engineering for Specialized Agents

6.1 ReAct: Reasoning and Acting

The **ReAct** (Reasoning + Acting) framework [15], introduced by Yao et al. 2023, is considered as a strong approach for building tool-using agents. ReAct works by alternating between reasoning steps and actions in a repeating **Thought → Action → Observation** loop.

The reasoning steps help the model plan, track progress, and adjust its strategy when an issue occurs. The action steps allow the agent to interact with external tools such as knowledge bases or environments to get more information. Results show that ReAct performs well, achieving a 34% improvement over imitation learning on ALFWorld tasks and a 10% improvement over reinforcement learning on WebShop benchmarks.

6.2 Chain-of-Thought Prompting

Chain-of-Thought (CoT) prompting [16] makes models reason through intermediate steps before giving a final answer. Wei at al. showed that with 8 CoT examples, a 540B-parameter model can achieve remarkable results on math reasoning tasks. The Layered Chain-of-Thought extension [17] breaks reasoning into separate layers. This is useful for multi-agent coordination.

6.3 Context Engineering for Agents

Anthropic's research on context engineering [18] goes further than basic prompt engineering, especially for tasks that require long-term planning by agents. It defines four strategies: write (create context), select (filter relevant information), compress, and isolate (separate concerns). System prompts should be organized into clear sections using either XML tags or Markdown headers. Getting the right balance between too specific and too vague instructions is important.

Studies on persona prompting show important limitations. Giving a simple role like "You are a travel expert." does not consistently improve performance. It is more effective to give detailed specifications for each specialized agent.

7. Benchmarking and Evaluation Metrics

7.1 MultiAgentBench Framework

MultiAgentBench [19], introduced at ACL 2025, is the first benchmark specifically designed to evaluate how well LLM-based agents collaborate. It defines several evaluation metrics:

- Task Score (TS): Uses LLM-based detection to check if the agents hit their targets and milestones
- Coordination Score (CS): Evaluates planning quality
- Communication Scores: Analyse the structure and effectiveness of message sending
- Topology Comparison: Compares different structures like star, chain, tree, and graph structures

The benchmark shows that graph-based coordination structures perform best. However, too many iterations, more than seven, reduce performance because of the communication noise it adds. This gives us an insight into how we should design the feedback loops between the Policy Compliance Agent and the booking agents.

7.2 Metrics for This Project

Based on the literature and sources, these metrics will be used to evaluate the multi-agent travel planning system:

- Task Completion Rate: I will look at binary success but also tracking milestones. It will be about seeing how far the agents can get.
- Coordination Score: Here we will see if the agents are truly collaborating or just ignoring each other.
- Message Exchange Count: I will keep track of the total number of messages it takes to get to a successful trip for keeping things as efficient as possible.
- Iteration Rounds: This will be to see how many back-and-forth rounds it takes between the Policy Agent and the booking agents.

8. Conclusion

This literature supports building a multi-agent business trip planning system. There are three main findings that will decide how the architecture will be configured.

First, hybrid architectures will be necessary. Combining LLM for reasoning and specialized agents for guardrails leads to the best performance. Studies show success rates of up to 97%, compared to just 0.6% for standalone LLMs on complex reasoning tasks. Pure LLM approaches struggle with constraints. Specialized agents provide the reliability needed for a business trip.

Additionally, coordination architecture is important. Centralized orchestration outperforms decentralized approaches. However, using too many feedback loops, more than seven, results in worse performance. This will help in designing the structure of the agent workflows.

And last, assigning simple personas to agents will not be enough. Techniques such as ReAct prompting and Chain-of-Thought reasoning will be necessary. Context engineering for long-term workflows is also a best practice that is commonly used.

LangGraph will be used for workflow orchestration and agent design. After reviewing multiple sources and methods, LangGraph comes out as the best option for this project. It’s the most robust and suitable option. Ollama will be used for local inference of the LLM. Ollama gives privacy, local inference and zero API costs.

Finally, evaluation will follow MultiAgentBench-methodology. Using task completion and coordination scores will be useful to demonstrate the benefits of a multi-agent approach.

9. References

[1] A. S. Rao and M. P. Georgeff, "BDI Agents: From Theory to Practice", 1995 Online available: https://cdn.aaai.org/ICMAS/1995/ICMAS95-042.pdf

[2] K.-T. Tran *et al.*, "Multi-Agent Collaboration Mechanisms: A Survey of LLMs", Jan. 2025. Online available: https://arxiv.org/pdf/2501.06322

[3] R. G. Smith, "The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver", Dec. 1980. Online available: https://www.reidgsmith.com/The_Contract_Net_Protocol_Dec-1980.pdf

[4] An Introduction to FIPA Agent Communication Language: Standards for Interoperable Multi-Agent System. Online available: https://smythos.com/developers/agent-development/fipa-agent-communication-language/

[5] T. Guo *et al.*, "Large Language Model based Multi-Agents: A Survey of Progress and Challenges", April 2024. Online available: https://arxiv.org/pdf/2402.01680

[6] Z. Chen *et al.*, "A Survey on LLM-based Multi-Agent System: Recent Advances and New Frontiers in Application", Dec. 2024. Online available: https://arxiv.org/pdf/2412.17481

[7] LangChain, "LangGraph Documentation," 2024. Online available: https://docs.langchain.com/langgraph/

[8] Q. Wu *et al.*, "AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation Framework", 2023. Online available: https://arxiv.org/pdf/2308.08155

[9] J. Xie *et al.*, "TravelPlanner: A Benchmark for Real-World Planning with Language Agents", 2024. Online available: https://arxiv.org/pdf/2402.01622

[10] A. Chen *et al.*, "TravelAgent: An AI Assistant for Personalized Travel Planning", Sep. 2024. Online available: https://arxiv.org/pdf/2409.08069

[11] "TRIP-PAL: Travel Planning with Guarantees by Combining Large Language Models and Automated Planners", *J.P. Morgan AI Research*, 2024. Online available: https://arxiv.org/pdf/2406.10196

[12] Y. Hao et al., "Large Language Models Can Solve Real-World Planning Rigorously with Formal Verification Tools", Apr. 2024. Online available: https://arxiv.org/pdf/2404.11891

[13] Ollama, "Ollama Documentation" 2024. Online available: https://ollama.com/

[14] G. Gerganov, "llama.cpp: LLM Inference in C/C++", GitHub repository. Online available: https://github.com/ggerganov/llama.cpp

[15] S. Yao *et al.*, "ReAct: Synergizing Reasoning and Acting in Language Models", 2023. Online available: https://arxiv.org/pdf/2210.03629

[16] J. Wei *et al.*, " Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, 2023. Online available: https://arxiv.org/pdf/2201.11903

[17] M. Sanwal, "Layered Chain-of-Thought Prompting for Multi-Agent LLM Systems", Jan. 2025. Online available: https://arxiv.org/pdf/2501.18645

[18] Anthropic, "Effective Context Engineering for AI Agents," Anthropic Engineering Blog.
Online available: https://www.anthropic.com/engineering/

[19] K. Zhu *et al.*, "MultiAgentBench: Evaluating the Collaboration and Competition of LLM Agents", 2025. Online available: https://arxiv.org/pdf/2503.01935