COALESCE: Economic and Security Dynamics of Skill-Based Task Outsourcing Among Team of Autonomous LLM Agents

Manish Bhatt[1+]
Researcher
OWASP/Project Kuiper Security
manish.bhatt13212@gmail.com

Ronald F. Del Rosario[2]
SAP ISBN Product Security
SAP
ron.del.rosario@sap.com

Vineeth Sai Narajala[3]
Proactive Security
Amazon Web Services
vineeth.sai@owasp.org

Idan Habler[3]
Adversarial AI Security reSearch
Intuit
idan_habler@intuit.com

***Abstract*—The meteoric rise and proliferation of autonomous Large Language Model (LLM) agents promise significant capabilities across various domains. However, their deployment is increasingly constrained by substantial computational demands, specifically for Graphics Processing Unit (GPU) resources. The high operational costs and resource requirements associated with training and inference for large-scale models hinder widespread adoption and the execution of complex, resource-intensive tasks. This paper addresses the critical problem of optimizing resource utilization in LLM agent systems. We introduce COALESCE (Cost-Optimized and Secure Agent Labour Exchange via Skill-based Competence Estimation), a novel framework designed to enable autonomous LLM agents to dynamically outsource specific subtasks to specialized, cost-effective third-party LLM agents. The framework integrates mechanisms for hybrid skill representation, dynamic skill discovery, automated task decomposition, a unified cost model comparing internal execution costs against external outsourcing prices, simplified market-based decision-making algorithms, and a standardized communication protocol between LLM agents. Comprehensive validation through 239 theoretical simulations demonstrates 41.8% cost reduction potential, while large-scale empirical validation across 240 real LLM tasks confirms 20.3% cost reduction with proper epsilon-greedy exploration, establishing both theoretical viability and practical effectiveness. The emergence of proposed open standards like Google's Agent2Agent (A2A) protocol further underscores the need for frameworks like COALESCE that can leverage such standards for efficient agent interaction. By facilitating a dynamic market for agent capabilities, potentially utilizing protocols like A2A for communication, COALESCE aims to significantly reduce operational costs, enhance system scalability, and foster the emergence of specialized agent economies, making complex LLM agent functionalities more accessible and economically viable.**

***Index Terms*—Agentic Applications, Agents, Large Language Models, Agent2Agent Protocol, Agent Security, AI Security, Graphics Processing Unit, Cloud Service Providers, Distributed Computing, Emerging Markets, Economics**

[1+]This work is not related to the author's position at Amazon. https://github.com/mbhatt1/COALESCE

[2]This work is not related to the author's position at SAP

[3]This work is not related to the author's position at Amazon Web Services.

[3]This work is not related to the author's position at Intuit

I. Introduction

A. The Rise of LLM Agents and Resource Challenges

Recent years have witnessed remarkable advancements in Large Language Models (LLMs), which exhibit significant potential for human-like intelligence, reasoning, and planning. These capabilities have spurred the development of LLM-based autonomous agents: systems designed to perceive environments, make decisions, and execute complex, multi-step tasks autonomously. These agents can leverage external tools, interact with diverse environments, and decompose problems [1]. Frameworks like LangChain and AutoGen [2] provide programming interfaces and structures for building these sophisticated agentic applications.

However, the operational deployment of these powerful agents faces significant hurdles, primarily stemming from their immense computational requirements [3]. LLMs, especially foundation models with billions or even hundreds of billions of parameters, demand substantial memory (VRAM) and processing power, particularly from GPUs, for both training and inference [4]. For instance, loading a model like GPT-3 (175B parameters) requires approximately 350 GB of VRAM even using a half-precision (FP16) format, while Llama-3-70B requires 140 GB [4]. Beyond model weights, significant memory is consumed by the Key-Value (KV) cache during inference, which scales with sequence length and the number of concurrent requests, potentially adding tens or hundreds of gigabytes to the requirement [4]. Specialized operations like Retrieval-Augmented Generation (RAG) [5] or fine-tuning further amplify these demands [4], [6].

These escalating resource demands present not only technical barriers but also significant economic challenges [4]. The cost of acquiring and operating high-end GPUs (like the Nvidia A100/H100 [3]) is substantial. Cloud-based GPU instances, while offering flexibility, incur significant operational expenses based on usage time [7]. For example, a single GPT-3 inference might cost between $0.0002 and $0.0014 in raw compute on an A100, translating to considerable expense at scale [8]. This high Total Cost of Ownership (TCO), encompassing hardware, software, and operations [9],

makes it economically unviable for many organizations or individual agents to maintain the peak infrastructure needed for all potential tasks. Consequently, resource constraints limit the scalability of agent deployments, the complexity of tasks they can undertake, and the accessibility of advanced AI capabilities [4].

B. Problem Statement: Optimizing Agent Operations via Outsourcing

The central problem addressed in this paper is how to enable the effective and scalable deployment of autonomous LLM agents while mitigating the prohibitive costs associated with their resource requirements. Specifically, how can an agent leverage specialized computational capabilities, such as intensive GPU processing for tasks like large-scale RAG or model fine-tuning, without incurring the full cost of owning and maintaining the necessary infrastructure?

This economic pressure naturally favors specialization and exchange, mirroring the evolution of cloud computing, where users access resources on demand [7]. We propose that inter-agent task outsourcing offers a viable solution. Using this paradigm, an agent facing a resource-intensive subtask can delegate its execution to other agents that possess the necessary specialized resources or skills and can perform the task more cost-effectively. This draws parallels with established practices like Business Process Outsourcing (BPO) and the use of cloud services for offloading computation [9]. Preliminary concepts exist, such as LLMs implicitly outsourcing knowledge retrieval via RAG [5], but a dedicated framework for active, cost-driven, and secure task outsourcing between autonomous agents is lacking. The recent introduction of open protocols like Google's Agent2Agent (A2A) [10], designed specifically to facilitate communication between agents from diverse vendors and frameworks, highlights the growing need and potential for structured inter-agent collaboration and outsourcing.

C. Proposed Framework: COALESCE

To address this need, we introduce COALESCE (Cost-Optimized Agent Labor Exchange via Skill-based Competence Estimation), a framework that provides a structured approach for autonomous LLM agents to make dynamic decisions about outsourcing subtasks to other agents within a multi-agent system or an open market, potentially leveraging proposed standards like A2A [10] for underlying communication.

The core principle of COALESCE is to enable a client agent, upon decomposing a larger task [1], to identify subtasks that are computationally expensive or require specialized skills it lacks. The client agent can then discover potential contractor agents possessing the requisite capabilities [11] and evaluate the trade-off between executing the subtask locally versus outsourcing it. This evaluation is based on a comprehensive cost model that considers not only the monetary price quoted by the contractor but also factors like latency, data transfer costs, integration overhead, and the inherent risks associated with relying on another LLM agent [7]. The decision to outsource is driven by a combination of skill compatibility and overall cost-efficiency.

D. Contributions

This paper makes the following contributions:

- **A Novel Framework (COALESCE):** We propose and formalize COALESCE, a framework specifically designed for dynamic, skill- and cost-driven secure task outsourcing between autonomous LLM agents, addressing the unique challenges posed by their computational needs and capabilities.
- **Unified Cost Model:** We define a comprehensive cost model that integrates internal resource consumption metrics (compute, time, API costs) with external market factors (contractor price, latency, data transfer, integration overhead, risk).
- **Integrated Decision-Making:** We outline a decision-making algorithm within COALESCE that incorporates both skill matching (using hybrid skill representation) and economic trade-off analysis for selecting optimal outsourcing opportunities.
- **Comprehensive Validation Framework:** We provide robust theoretical validation through 239 mathematical simulation runs demonstrating 41.8% ± 10.5% average cost reduction, combined with large-scale empirical validation using 240 actual LLM tasks across 4 task types (GPT-4 and Claude-3.5-Sonnet) achieving 20.3% cost reduction with proper epsilon-greedy exploration, confirming both the mathematical framework's potential and the critical importance of exploration mechanisms in real-world deployment.
- **Critical Exploration Mechanism Insights:** We demonstrate through real agent validation that epsilon-greedy exploration is not merely a theoretical optimization but an essential requirement for practical performance, with exploration failure reducing cost reduction from 20.3% to only 1.9% in real-world scenarios. This finding reveals fundamental limitations in the deterministic decision algorithm and provides clear directions for architectural improvements.
- **Algorithmic Limitation Analysis:** We identify and analyze the exploration dependency as a critical system weakness, proposing specific mitigation strategies including adaptive thresholds, market-maker architectures, and advanced exploration algorithms to achieve robust performance without reliance on random exploration mechanisms.

II. Related Work

This section surveys existing research relevant to the COALESCE framework, spanning autonomous LLM agents, multi-agent systems (MAS) with a focus on task allocation and economic mechanisms, skill representation and discovery, economic models in distributed computing, and task decomposition techniques for LLMs.

A. Autonomous LLM Agents

LLM-based autonomous agents leverage the capabilities of LLMs to perform complex tasks requiring reasoning, planning,

and interaction with external environments or tools [1]. Frameworks such as LangChain, AutoGen [2], and AgentSquare provide architectures often comprising modules for profiling (defining agent roles), memory (handling context limits [1]), planning (task decomposition and action sequencing), and action execution (interacting with tools or APIs) [1]. Agents can use techniques like Retrieval Augmented Generation (RAG) [5] or specialized tools to enhance their knowledge and capabilities [1].

Despite their potential, LLM agents face significant limitations. They exhibit inherent unpredictability and can produce "hallucinations" or factually incorrect outputs [1]. Uncertainty can propagate and compound in multi-step tasks or multi-agent interactions, potentially compromising system stability and correctness [1]. Context window limitations [1] hinder performance on tasks requiring long-term memory, although memory modules aim to mitigate this [1]. Agents can struggle with long-horizon planning [1], prompt robustness (sensitivity to input phrasing) [1], and maintaining alignment with human values or task specifications [5]. Furthermore, significant trustworthiness and safety concerns arise, particularly in multi-agent settings where vulnerabilities like prompt injection or knowledge poisoning can occur [12]. The high cost and lack of transparency of proprietary models like GPT-4 motivate research into optimizing open-source LLMs for agentic tasks [1], though open models often lag in agent-specific capabilities [1].

B. Multi-Agent Systems (MAS)

MAS involves multiple autonomous agents interacting within a shared environment to achieve individual or collective goals [13]. Agents can be cooperative, competitive, or have mixed motives [14]. MAS offers parallelism, robustness, and scalability for complex, distributed problems [15]. Key challenges in MAS include coordination, communication, and task allocation [13].

1) Task Allocation: Assigning tasks to agents is a fundamental problem, often NP-hard in the general case [15]. Objectives typically involve minimizing execution time or cost, maximizing the number of completed tasks, or maximizing the overall system utility [15]. Allocation can be static (pre-assigned) or dynamic (assigned during runtime, offering more robustness) [15]. Methodologies can be centralized (a single entity makes assignments) or decentralized (agents negotiate or bid) [15].

2) Contract Net Protocol (CNP): A well-known decentralized task allocation protocol [16]. It involves a manager agent broadcasting a task announcement (call-for-proposals), contractor agents submitting bids (proposals), the manager selecting a winner (accept/reject), and the winner executing and reporting back (inform/cancel) [16]. CNP allows finding suitable agents based on their proposals and is standardized by FIPA [17]. However, it can suffer from significant communication overhead and network congestion, especially in large systems [16]. Its basic form may not be optimal [16], [18], lacks mechanisms for managers to specify preferences beyond the proposal content, and handles task failures simplistically [16]. Extensions have been proposed, such as using buffer pools [19], setting bidding thresholds [19], restricting the audience for announcements [16], or employing two-stage procedures [19]. While established protocols like CNP offer interaction frameworks, applying them directly to LLM agents necessitates addressing the unique nature of LLM skills, which are often qualitative and probabilistically executed, potentially complicating proposal evaluation and contract fulfillment [1].

3) Auction Mechanisms: Auctions provide formal protocols for allocating resources or tasks based on bids, determining both winners and payments [16]. Various types exist, including single-good, multi-unit, and combinatorial auctions where agents bid on bundles of items [16].

Double Auctions (DA): Allow buyers and sellers to submit bids and asks simultaneously. A market-clearing price is determined, facilitating trade between buyers bidding above and sellers asking below this price [16]. DAs are used in stock exchanges and other two-sided markets [20]. They can be analyzed using game theory [21].

Vickrey-Clarke-Groves (VCG) Mechanisms: A class of truthful mechanisms where bidders are incentivized to bid their true valuations [22]. Winners pay an amount based on the externality (harm or benefit) their participation imposes on others [22]. VCG maximizes social welfare (sum of true values) but may not be budget-balanced (auctioneer might need to subsidize or make a profit) and can be vulnerable to collusion [22].

Key properties of auction mechanisms include [23]:

- Efficiency (EE): Allocating goods/tasks to maximize total value or social welfare.
- Incentive Compatibility (IC) / Truthfulness: Ensuring agents' best strategy is to reveal their true valuations (Dominant Strategy IC - DSIC, or Nash Equilibrium IC - NEIC).
- Individual Rationality (IR): Participation is beneficial for agents (non-negative utility).
- Budget Balance (BB): Ensuring the auctioneer breaks even (Strong BB) or does not lose money (Weak BB).

The Myerson-Satterthwaite theorem shows it's impossible to achieve EE, BB, IR, and IC simultaneously in general bilateral trade settings [24]. Mechanisms like McAfee's DA achieve truthfulness and BB by sacrificing some efficiency (e.g., dropping one potential trade) [21]. Similar to CNP, applying standard auction theory to LLM agents requires careful consideration of how to define and bid on tasks with uncertain outcomes and complex, non-scalar "skills" [1].

4) Agent2Agent (A2A) Protocol: Introduced by Google in April 2025 [10], A2A is an open protocol designed specifically to enable seamless communication and collaboration between AI agents, regardless of their vendor or underlying framework. It aims to solve the enterprise integration challenge of agent interoperability by acting as a "universal translator." A2A is built on existing web standards like HTTP and JSON-RPC and defines mechanisms for capability discovery via "Agent Cards" (JSON files describing skills, endpoints, etc.), task management with defined lifecycle states, agent-to-agent

collaboration through context sharing, and negotiation of user experience modalities (text, audio, video). A2A operates on a client-server model where a client agent initiates tasks with a remote agent. It is positioned as complementary to protocols like Anthropic's Model Context Protocol (MCP), with MCP focusing on agent-tool interaction [25] (vertical integration) and A2A focusing on agent-agent interaction (horizontal integration). A2A aims to enable dynamic "digital workforce teams" and lower integration costs for multi-agent systems.

5) MAS Challenges: Beyond allocation, MAS faces challenges like achieving mutual understanding [1], managing uncertainty propagation [1], ensuring robust communication [13], and dealing with non-stationarity where agents' policies change concurrently during learning [26]. Recent studies on LLM-based MAS identify specific failure modes [27], including poor specification following, inter-agent misalignment (e.g., ignoring input, withholding information), and problems with task verification and termination [2]. These highlight the difficulties in ensuring reliable collaboration, particularly when individual agent behavior is already unpredictable [2]. Protocols like A2A [10] aim to mitigate some communication challenges but do not inherently solve issues of agent alignment or task verification.

C. Skill Representation & Discovery in MAS

For effective collaboration and task allocation, agents need mechanisms to represent and discover each other's capabilities or skills [1].

1) Explicit Representation: One approach uses formal, predefined structures. Ontologies, such as OASIS [28], provide a semantic framework to define agent behaviors, capabilities, goals, and commitments using languages like OWL [28]. Agents commit to an ontology, enabling shared understanding and interoperability [28]. Standards bodies like FIPA have also defined agent communication languages and interaction protocols (including CNP) [17]. Rule-based systems can implicitly encode capabilities [29]. Google's A2A protocol [10] introduces "Agent Cards" as a standardized JSON format for agents to advertise their capabilities, endpoint URLs, and authentication requirements, facilitating discovery. While providing clarity and structure, these explicit methods, including Agent Cards, can be rigid and may struggle to capture the full spectrum of nuanced or emergent skills, particularly in rapidly evolving systems like LLMs [28].

2) Implicit/Learned Representation: Alternatively, skills can be learned from experience. In MARL, skill discovery techniques aim to learn latent representations (skills) that capture useful temporal abstractions or behavioral patterns without explicit rewards [30]. Methods often maximize mutual information between skills and states/trajectories [30]. Hierarchical approaches like HMASD [30] learn both team-level and individual-level skills concurrently. Architectures like ALMA [11] use learned agent embeddings, derived from agent states and interactions, within the allocation policy. The allocator learns to map agent embeddings (implicitly representing capabilities) to subtask embeddings to optimize assignments [11]. These learned representations offer flexibility and can capture complex, emergent capabilities, but often lack interpretability and a direct link to economic value needed for market mechanisms. A key challenge remains in bridging these functional, learned skills with communicable, verifiable, and economically priceable attributes for negotiation protocols [30].

D. Economic Models in Distributed Cloud Computing

The rise of cloud computing fundamentally shifted distributed systems design by introducing pricing as a primary interface between users (consuming resources) and providers (supplying resources) [7]. This necessitates considering economic factors alongside traditional performance metrics.

1) Cost Components: Evaluating the cost of computation involves more than just the sticker price. Total Cost of Ownership (TCO) includes capital expenditures (hardware) and operational expenditures (software licenses, energy, administration, maintenance) [9]. Operations costs often dominate [9]. Cloud pricing typically follows a pay-as-you-go model based on resource consumption, such as virtual machine instance hours, CPU time, storage used, data transferred (often with egress fees), and API calls [7]. Compute cost for LLMs can be roughly estimated based on FLOPs, which depend on model size (parameters) and sequence length [8]. Memory requirements (weights, KV cache, activations) also translate to cost, as they dictate the necessary hardware tier [4]. Hidden or transaction costs, such as those for migration, change management, integration, and dealing with vendor lock-in, are also significant but often underestimated [31].

2) Pricing Models: Cloud providers utilize various pricing strategies. Pay-as-you-go offers flexibility [7]. Subscription models provide fixed prices for longer commitments [7]. Reserved instances offer discounts for capacity commitments [7]. Spot markets allow bidding on spare capacity at potentially lower, but variable and interruptible, prices [7]. There is a trend towards more dynamic pricing models to better match supply and demand and utilize resources efficiently [7].

3) Agent-Based Economics: Agent technology is applied in cloud and distributed environments for tasks like automated negotiation of Service Level Agreements (SLAs), dynamic resource allocation, service composition, and elasticity management [32]. Agent-Based Computational Economics (ACE) uses computational models of interacting agents to study economic phenomena [33]. Agent marketplaces facilitate the buying and selling of resources or services [32].

4) Challenges: Estimating the true TCO of cloud adoption remains difficult due to hidden costs [31]. Ensuring pricing fairness between providers and users is complex [7]. The cloud market exhibits significant concentration among a few large providers (AWS, Azure, GCP), raising concerns about competition and switching costs [34]. Managing the cost implication of system failures is also crucial [7].

E. Task Decomposition & Planning for LLMs

Executing complex, multi-step tasks effectively requires agents to decompose them into smaller, manageable subtasks

[1]. This mirrors human problem-solving and algorithmic challenges like "divide and conquer" [35].

1) Approaches: LLMs themselves can be prompted to perform task decomposition [36]. Some frameworks decompose tasks first and then plan for each subtask, while others interleave decomposition and planning/execution [35]. Hierarchical planning methods, like Meta-Task Planning (MTP) or Planning with Multi-Constraints (PMC) [35], break tasks into subordinate task hierarchies. Frameworks like TDAG [37] dynamically decompose tasks and generate specialized subagents for each subtask. Hybrid approaches combine LLMs for high-level decomposition or commonsense reasoning with classical planners (e.g., PDDL-based) for generating guaranteed executable action sequences for sub-goals [36]. Research also explores generating and optimizing entire workflows using LLMs.

2) Challenges: A key challenge is error propagation where errors made during the execution of one subtask can negatively impact subsequent steps [37]. Ensuring the generated sub-plans are valid and lead to the overall goal requires robust planning and potentially reflection or verification mechanisms [35].

F. Gap Analysis & Motivation for COALESCE

The reviewed literature reveals significant progress in individual areas but highlights a gap at their intersection. MAS research offers mature task allocation protocols like CNP [16] and auction mechanisms [22], but these often assume predictable agents and well-defined, easily quantifiable tasks or skills—assumptions challenged by the probabilistic nature and complex capabilities of LLMs [1]. The emergence of protocols like A2A [10] provides a potential standard for agent-agent communication but does not inherently include mechanisms for economic negotiation or cost-based decision-making for task allocation. LLM agent research primarily focuses on enhancing single-agent reasoning, planning, and tool use [1], or on collaborative frameworks that don't explicitly model market dynamics and cost optimization for resource-intensive tasks. While economic models for cloud computing exist [7], they don't specifically address the unique cost structures and skill-based value propositions of delegating cognitive tasks between LLM agents.

COALESCE aims to bridge this gap by proposing a framework tailored for LLM agents. It integrates concepts from MAS task allocation (market-based negotiation logic), skill representation (hybrid ontology/learned approach, potentially leveraging A2A Agent Cards [10]), LLM agent planning (task decomposition), and cloud economics (explicit cost modeling including compute, time, risk, and price) to enable efficient, dynamic outsourcing driven by both capability matching and resource optimization needs, particularly targeting high-cost resources like GPUs. COALESCE could potentially utilize A2A [10] as the underlying communication layer, adding the necessary economic decision-making layer on top.

III. Proposed Framework: COALESCE (Cost-Optimized Agent Labor Exchange via Skill-based Competence Estimation)

A. Overview

COALESCE is conceptualized as a decentralized framework enabling autonomous LLM agents to optimize resource utilization and operational costs by strategically outsourcing subtasks to other agents. The framework operates on a hybrid architecture combining peer-to-peer discovery mechanisms with optional centralized reputation services, implementing a multi-layered security model with cryptographic verification protocols.

1) System Architecture: The COALESCE framework implements a modular architecture consisting of five core components:

1) **Agent Discovery Layer (ADL):** Implements a distributed hash table (DHT) based on Kademlia protocol for decentralized agent discovery, with fallback to centralized registries [38]. Each agent maintains a routing table of up to 160 entries per k-bucket, enabling $O(\log n)$ lookup complexity for agent discovery across the network.
2) **Skill Verification Engine (SVE):** Utilizes a combination of zero-knowledge proofs for resource verification and benchmark-based skill attestation. Implements a Merkle tree structure for skill certificates with SHA-256 hashing, enabling efficient verification of agent capabilities without revealing proprietary information.
3) **Economic Decision Module (EDM):** Employs a multi-criteria decision analysis (MCDA) framework using the Technique for Order Preference by Similarity to Ideal Solution (TOPSIS) algorithm, weighted by dynamic market conditions and historical performance data.
4) **Secure Communication Protocol (SCP):** Implements end-to-end encryption using Elliptic Curve Diffie-Hellman (ECDH) key exchange with AES-256-GCM for message encryption, ensuring confidentiality and integrity of task data during outsourcing operations.
5) **Reputation and Trust Management (RTM):** Utilizes a blockchain-based reputation system with Practical Byzantine Fault Tolerance (pBFT) consensus, maintaining tamper-resistant records of agent performance with exponential decay functions for temporal relevance weighting.

The framework defines two primary roles:

- **Client Agent:** An LLM agent that needs to accomplish a task. It decomposes the task, identifies potential subtasks for outsourcing (especially those requiring resources it lacks or finds expensive, e.g., high-end GPUs), evaluates the cost-benefit trade-off, discovers suitable contractors, negotiates terms (or decides based on advertised terms), and manages the outsourcing process, potentially initiating tasks via A2A [10] [39].
- **Contractor Agent:** An LLM agent (or potentially a specialized non-LLM service wrapped as an agent) that possesses specific skills, resources (e.g., GPU capacity), or expertise. It advertises its capabilities (e.g., via A2A Agent Cards [10]),

evaluates task requests from clients, executes awarded tasks, and delivers results (potentially using A2A task management [10]).

A potential secondary role is the **Broker/Registry Agent**, which could facilitate the discovery process by maintaining a directory of contractor agents, their skills, and potential reputation scores, possibly interacting with or aggregating A2A Agent Cards [10].

The high-level workflow within COALESCE proceeds as follows:

1) **Task Reception & Decomposition:** The Client Agent receives a high-level task. It uses its planning module [1] to decompose the task into a sequence or graph of subtasks.
2) **Outsourcing Candidate Identification:** The Client identifies subtasks that are suitable candidates for outsourcing based on criteria such as high estimated local resource cost (e.g., GPU-intensive), requirement for specialized skills the Client lacks, or potential for parallel execution.
3) **Outsourcing Decision (Cost-Benefit Analysis):** For each candidate subtask, the Client estimates the cost of local execution versus the potential cost of outsourcing, considering skill requirements and risk (detailed in Sections III-E and III-F).
4) **Contractor Discovery:** If outsourcing is deemed potentially beneficial, the Client searches for suitable Contractor Agents using the skill discovery mechanism (Section III-C), potentially leveraging A2A Agent Cards [10].
5) **Negotiation/Selection:** Based on the cost-benefit analysis and skill matching, the Client selects the optimal Contractor. This step precedes formal task initiation via a protocol like A2A [10].
6) **Task Initiation & Execution:** The Client initiates the task with the selected Contractor (e.g., via A2A `tasks/send` [10]). The Contractor executes the subtask.
7) **Result Verification & Integration:** The Client receives the results (e.g., as an A2A Artifact [10]), verifies them against predefined criteria, and integrates them into its overall plan.
8) **Payment/Settlement:** Upon successful verification, the Client facilitates payment to the Contractor.

This is represented in the sequence diagram in Fig. 1.

B. Agent Skill Representation

To enable effective matching between task requirements and contractor capabilities, COALESCE utilizes a hybrid skill representation model, aiming to balance structure, verifiability, and the ability to capture nuanced LLM abilities. This hybrid approach attempts to balance the need for verifiable attributes with the ability to represent nuanced, emergent capabilities [30]. Ontologies provide a common ground for basic, checkable skills (like hardware access), while embeddings capture nuanced, harder-to-define capabilities learned through experience.

1) Ontology-Based Core Skills & Standardized Representation: A standardized, shared ontology (e.g., using OWL [28]) defines a vocabulary for common, relatively unambiguous, and potentially verifiable skills and resources. Google's A2A protocol [10] introduces "Agent Cards", a JSON format for agents to advertise capabilities, endpoint URLs, and authentication requirements. COALESCE could leverage Agent Cards as the standard format for representing these core, explicit skills, promoting interoperability. Examples mapped to potential Agent Card fields:

- `hasResource:GPU_Type` (e.g., NVIDIA_A100_80GB) → Could be listed under agent skills or capabilities in the Agent Card JSON.
- `hasAccessTo:API` (e.g., PubMed_API, Weather_API) → Could be listed under skills.
- `canExecute:SoftwareLibrary` (e.g., PyTorch, TensorFlow, vLLM_Runtime [6]) → Could be listed under skills.
- `hasKnowledgeBase:Domain` (e.g., Medical_Literature, Legal_Case_Law) → Could be listed under skills.
- `supportsProtocol:CommunicationStandard` (e.g., A2A_v1.0) → Implicit if using A2A [10], or explicitly stated.

This allows for efficient filtering and basic compatibility checks using a standardized mechanism [28].

2) Learned Skill Embeddings: For more complex, qualitative, or emergent capabilities (e.g., "proficient in generating concise technical summaries," "adept at empathetic dialogue simulation," "high accuracy in RAG for financial documents"), agents can utilize learned vector embeddings. These embeddings capture nuances potentially beyond the structured Agent Card format and can be used for similarity-based matching during discovery and selection [11]. The Agent Card format might potentially be extended or used alongside embeddings, perhaps by including references or hashes to embedding models within the card's metadata. Embeddings can be generated via:

- Training on specific tasks or interaction data [30].
- Fine-tuning foundation models on agent-specific datasets [1].
- Self-assessment outputs from the LLM itself, potentially mapped to a shared embedding space.

3) Self-Reported Profiles: Each agent maintains a profile that combines these elements. It explicitly declares skills via its Agent Card [10], provides pointers or hashes to its relevant skill embeddings (potentially within the card or separately), and may include additional metadata such as performance benchmarks, resource availability schedules, and initial cost parameters (which might be included in the Agent Card or obtained through initial interaction) [1].

C. Skill Discovery Mechanism

Clients need a way to find contractors matching their subtask requirements. COALESCE can leverage the discovery mechanisms inherent in protocols like A2A [10], alongside other methods.

- **A2A Agent Card Discovery:** This becomes the primary mechanism if A2A [10] is adopted. Clients fetch the Agent

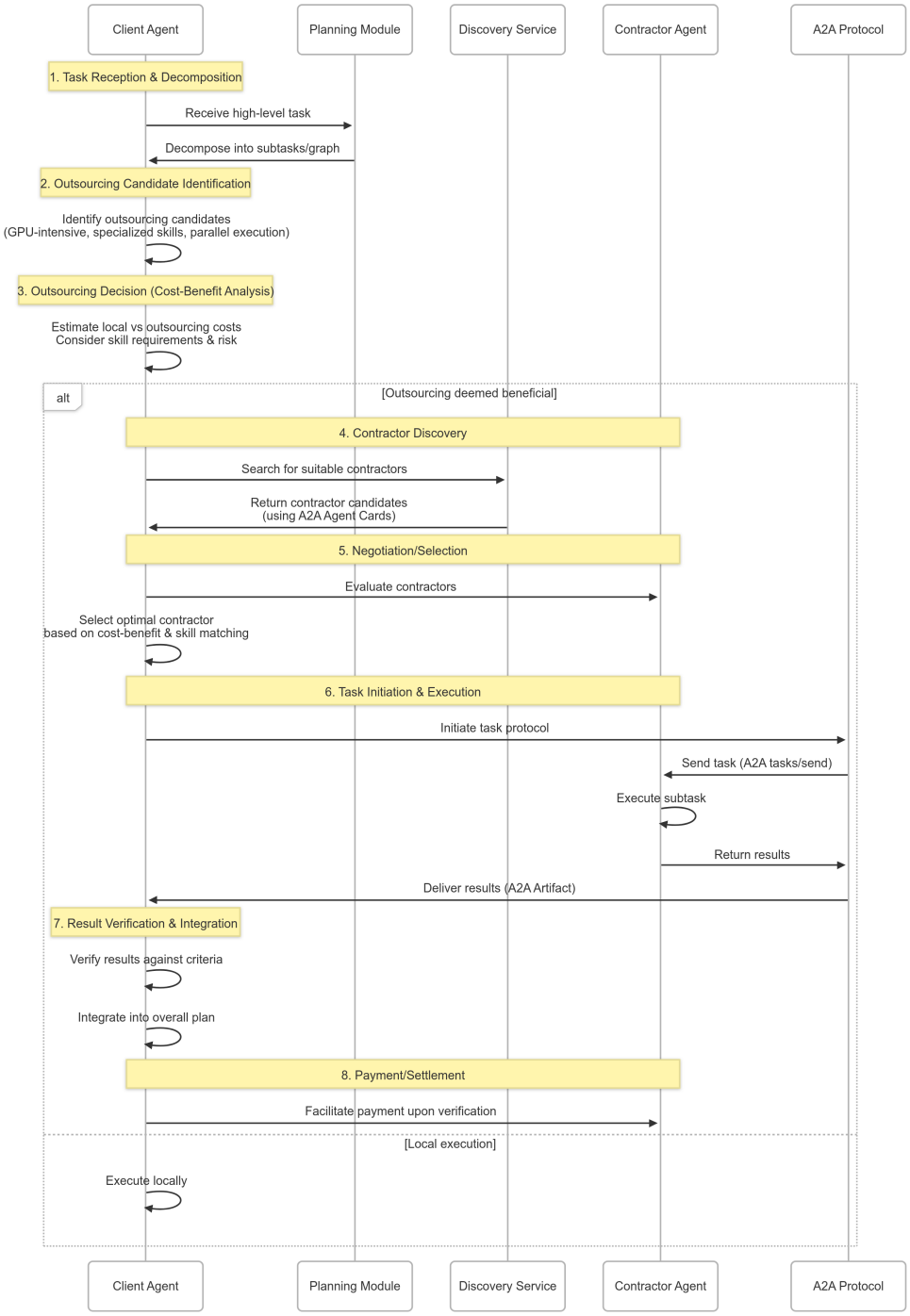

Fig. 1. High-level workflow within COALESCE.

Card JSON file, typically located at a well-known URL (`/.well-known/agent.json`) for a potential contractor agent. The client parses the card to determine the agent's capabilities, endpoint, authentication needs, and supported protocols. This allows for standardized, direct discovery without necessarily relying on a central registry [40].

- **Registry-Based Discovery:** A central or distributed registry could still exist, potentially aggregating Agent Cards or providing pointers to them. Clients query this registry with their task requirements (specified using the ontology/embedding characteristics) to receive a list of potential candidates and their Agent Card locations. This offers efficiency for broad searches but relies on the registry's availability and integrity.
- **Peer-to-Peer (P2P) Discovery / Targeted Probing:** While

A2A [10] focuses on direct client-server interaction based on known endpoints (often found via Agent Cards), P2P broadcast mechanisms (like in CNP [16]) could still be used in scenarios where potential contractors are unknown. Alternatively, a client might probe known agents by directly requesting their Agent Cards.

D. Task Decomposition & Requirement Specification

Effective outsourcing requires the Client Agent to clearly define the subtask and its requirements.

- **Task Decomposition:** The Client utilizes its internal planning module, potentially leveraging the LLM's reasoning capabilities [36] or hierarchical planning techniques [35], to break the overall goal into subtasks. It identifies which subtasks are candidates for outsourcing based on resource intensity or skill gaps [35].
- **Requirement Specification:** For each subtask T to be potentially outsourced, the Client generates a formal specification document. This document serves as the basis for discovery (matching against Agent Cards/profiles) and task initiation (forming the content of the initial A2A message [10]). It should include:
  - *Required Skills:* References to specific terms in the shared skill ontology (expected in Agent Cards [10]) and/or target characteristics described via natural language or embedding vectors.
  - *Resource Needs:* Estimated computational load (e.g., target FLOPs [8]), minimum memory requirements [4], required software libraries or tools [1].
  - *Input Data:* Description of the input data format, size, and method of access (potentially passed as A2A Message Parts [10]).
  - *Output Requirements:* Desired format and structure of the results (expected as an A2A Artifact [10]).
  - *Verification Criteria:* Measurable criteria for validating the correctness or quality of the delivered output.
  - *Constraints:* Hard constraints such as maximum acceptable latency, maximum budget (price), required data privacy/security level [41], geographical restrictions, etc.

E. Cost Modeling

COALESCE implements a comprehensive multi-dimensional cost modeling framework that captures both direct and indirect costs associated with task execution. The model incorporates real-time market dynamics, resource utilization patterns, and risk assessment metrics to enable precise economic decision-making.

1) Internal Cost ($C_{internal}$): The internal cost estimation employs a detailed resource consumption model that accounts for hardware utilization, energy consumption, and opportunity costs. The comprehensive formula is:

$$C_{\text{internal}}(T) = C_{\text{compute}} + C_{\text{memory}} + C_{\text{energy}} + C_{\text{opportunity}} + C_{\text{depreciation}} \tag{1}$$

Where each component is calculated as follows:

Compute Cost (C_{compute}): Based on FLOPS estimation and hardware specifications:

$$C_{\text{compute}} = \frac{\text{FLOPS}(T)}{P_{\text{peak}}} \times t_{\text{exec}} \times C_{\text{hw\_hour}} \tag{2}$$

where $\text{FLOPS}(T)$ represents the floating-point operations required for task T, t_{exec} is the estimated execution time, P_{peak} is the peak performance of the local hardware, and $C_{\text{hw\_hour}}$ is the hourly cost of hardware utilization.

Memory Cost (C_{memory}): Accounts for VRAM and system memory utilization:

$$C_{\text{memory}} = \left(\frac{M_{\text{model}} + M_{\text{kv\_cache}} + M_{\text{activations}}}{M_{\text{total}}} \right) \times t_{\text{exec}} \times C_{\text{mem\_hour}} \tag{3}$$

where M_{model} is model weight memory, $M_{\text{kv\_cache}}$ is the Key-Value cache memory requirement, $M_{\text{activations}}$ is activation memory, and M_{total} is total available memory.

Energy Cost (C_{energy}): Incorporates power consumption patterns:

$$C_{\text{energy}} = P_{\text{TDP}} \times U_{\text{factor}} \times t_{\text{exec}} \times C_{\text{kwh}} \tag{4}$$

where P_{TDP} is the Thermal Design Power, U_{factor} is the utilization factor (0.7-0.95 for GPU-intensive tasks), and C_{kwh} is the cost per kilowatt-hour.

Opportunity Cost ($C_{\text{opportunity}}$): Quantifies the value of alternative tasks that could be executed:

$$C_{\text{opportunity}} = \max_{i \in Q} \left(\frac{V_i}{t_i} \right) \times t_{\text{exec}} \tag{5}$$

where Q is the queue of pending tasks, V_i is the value of task i, and t_i is its execution time.

2) External Cost ($C_{external}$): The external cost model incorporates a sophisticated risk assessment framework and dynamic pricing mechanisms:

$$\begin{aligned} C_{\text{external}}(T, A_j) = & P_j(T) + C_{\text{communication}} + C_{\text{verification}} \\ & + C_{\text{integration}} + C_{\text{risk}} + C_{\text{latency\_penalty}} \end{aligned} \tag{6}$$

Dynamic Pricing ($P_j(T)$): Contractor pricing based on supply-demand dynamics:

$$P_j(T) = P_{\text{base}} \times \left(1 + \alpha \times \frac{D_{\text{current}} - S_{\text{available}}}{S_{\text{total}}} \right) \times \beta_{\text{complexity}}(T) \tag{7}$$

where P_{base} is the base price, α is the demand sensitivity factor, D_{current} is current demand, $S_{\text{available}}$ is available supply, and $\beta_{\text{complexity}}(T)$ is the task complexity multiplier.

Communication Cost ($C_{\text{communication}}$): Accounts for data transfer and protocol overhead:

$$C_{\text{communication}} = \left(\frac{S_{\text{input}} + S_{\text{output}}}{B_{\text{bandwidth}}} \right) \times C_{\text{transfer}} + C_{\text{protocol\_overhead}} \tag{8}$$

where S_{input} and S_{output} are input and output data sizes, $B_{\text{bandwidth}}$ is available bandwidth, and $C_{\text{protocol\_overhead}}$ includes encryption and authentication costs.

Risk Cost (C_{risk}): Multi-factor **Risk Cost (C_{risk}):** Multi-factor risk assessment:

$$C_{\text{risk}} = V_{\text{task}} \times [1 - (1 - P_{\text{failure}})(1 - P_{\text{security}})(1 - P_{\text{quality}})] \tag{9}$$

where V_{task} is the task value, P_{failure}, P_{security}, and P_{quality} are failure, security breach, and quality degradation probabilities respectively, and γ_{impact} is the impact severity multiplier.

3) Real-time Cost Calibration: COALESCE implements an adaptive calibration mechanism using exponential weighted moving averages (EWMA) to adjust cost estimates based on historical performance:

$$\hat{C}_t = \lambda \times C_{\text{actual},t-1} + (1 - \lambda) \times \hat{C}_{t-1} \tag{10}$$

where $\hat{C}_t$ is the calibrated cost estimate at time t, $C_{\text{actual},t-1}$ is the actual cost from the previous execution, and $\lambda \in [0.1, 0.3]$ is the learning rate parameter.

F. Advanced Multi-Criteria Decision-Making Algorithm

COALESCE implements a sophisticated decision-making framework that combines multi-criteria decision analysis (MCDA) with machine learning-based prediction models and game-theoretic optimization principles.

1) Core Decision Algorithm: The decision process employs a hybrid approach combining TOPSIS (Technique for Order Preference by Similarity to Ideal Solution) with correlation-aware weight adjustment and dynamic market adaptation. To address TOPSIS's independence assumption, we implement correlation-adjusted weights using the formula:

$$w'_i = w_i \cdot (1 - \alpha \sum_{j \neq i} |\rho_{ij}|) \tag{11}$$

where w'_i is the correlation-adjusted weight, w_i is the base weight, $\alpha = 0.3$ is the correlation penalty factor, and ρ_{ij} is the Pearson correlation coefficient between criteria i and j.

```
Algorithm: COALESCE_Decision_Engine
Input: Task T, Candidates S_candidates,
    Market_State M, History H
Output: Decision D = {LOCAL, OUTSOURCE(A_j)}

// Phase 1: Multi-dimensional Cost Analysis
C_internal ← Calculate_Internal_Cost(T)
Criteria_Matrix ← Initialize_Criteria_Matrix()

For each A_j in S_candidates:
  If Skill_Compatibility_Check(A_j, T) ≥ θ_skill:
    C_external[j] ← Calculate_External_Cost(T,
    A_j)
    Reliability[j] ← Get_Reliability_Score(A_j,
    H)
    Latency[j] ← Estimate_Task_Latency(T, A_j)
    Security[j] ← Assess_Security_Risk(A_j, T)
    Criteria_Matrix[j] ← [C_external[j],
    Reliability[j], Latency[j], Security[j]]
  End If
End For

// Phase 2: Dynamic Weight Calculation
// Market-adaptive weight calculation using
    exponential decay
β ← 0.7  // Market responsiveness factor
w_cost ← β · Market_Pressure(M) + (1-β) ·
    Historical_Weight(H, "cost")
w_reliability ← β · Failure_Rate(M) + (1-β) ·
    Historical_Weight(H, "reliability")
w_latency ← β · Task_Urgency(T) + (1-β) ·
    Historical_Weight(H, "latency")
w_security ← β · Risk_Level(T) + (1-β) ·
    Historical_Weight(H, "security")
// Normalize weights to sum to 1
W ← Normalize([w_cost, w_reliability,
    w_latency, w_security])
// Apply correlation adjustment from Equation
    (1)
W ← Apply_Correlation_Adjustment(W,
    Criteria_Matrix)
// Phase 3: TOPSIS Multi-Criteria Analysis
Normalized_Matrix ←
    Normalize_Criteria_Matrix(Criteria_Matrix)
Weighted_Matrix ←
    Apply_Weights(Normalized_Matrix, W)
Ideal_Solution ←
    Calculate_Ideal_Solution(Weighted_Matrix)
Anti_Ideal ←
    Calculate_Anti_Ideal_Solution(Weighted_Matrix)

For each candidate A_j:
  D_positive[j] ← Euclidean_Distance(A_j,
  Ideal_Solution)
  D_negative[j] ← Euclidean_Distance(A_j,
  Anti_Ideal)
  TOPSIS_Score[j] ← D_negative[j] /
  (D_positive[j] + D_negative[j])
End For

// Phase 4: Game-Theoretic Optimization
A_best ← argmax(TOPSIS_Score)
Nash_Equilibrium ←
    Calculate_Nash_Strategy(C_internal,
    C_external[A_best])

// Phase 5: Final Decision with Confidence
    Interval
Confidence ← Calculate_Decision_Confidence(H,
    T, A_best)
If (TOPSIS_Score[A_best] > τ_threshold) AND
    (Confidence > ρ_min):
  Return OUTSOURCE(A_best)
Else:
  Return LOCAL
End If
```

Listing 1. COALESCE Advanced Decision Algorithm

2) Skill Compatibility Assessment: The skill compatibility check employs a hybrid semantic similarity approach combining ontological matching with embedding-based similarity:

$$\begin{aligned}\text{Skill\_Compatibility}(A_j, T) = \alpha \times S_{\text{ontological}} \\ + \beta \times S_{\text{embedding}} + \gamma \times S_{\text{performance}}\end{aligned} \tag{12}$$

where:

- $S_{\text{ontological}}$ is the ontological skill match score using Jaccard similarity on skill sets
- $S_{\text{embedding}}$ is the cosine similarity between task requirements and agent skill embeddings
- $S_{\text{performance}}$ is the historical performance score for similar tasks
- $\alpha + \beta + \gamma = 1$ with typical values $\alpha = 0.3, \beta = 0.5, \gamma = 0.2$

3) Dynamic Weight Calculation: The weight calculation mechanism adapts to market conditions and task characteristics using a reinforcement learning approach:

$$w_i^{(t)} = w_i^{(t-1)} + \eta \times \nabla_w Q(s_t, a_t, w_i^{(t-1)}) \tag{13}$$

where $Q(s_t, a_t, w_i)$ is the Q-function representing the expected utility of action a_t in state s_t with weight w_i, and η is the learning rate.

4) Reliability Score Calculation: The reliability assessment incorporates multiple factors using a Bayesian approach:

$$R(A_j) = \frac{\alpha_{\text{success}} + n_{\text{success}}}{\alpha_{\text{success}} + \beta_{\text{failure}} + n_{\text{total}}} \times e^{-\lambda \times t_{\text{decay}}} \tag{14}$$

where α_{success} and β_{failure} are Beta distribution parameters, n_{success} and n_{total} are observed successes and total interactions, and $e^{-\lambda \times t_{\text{decay}}}$ provides temporal decay with $\lambda = 0.1$ per month.

5) Security Risk Assessment: The security risk evaluation employs a multi-layered approach considering data sensitivity, agent reputation, and communication channel security:

$$\begin{aligned} \text{Security\_Risk}(A_j, T) = & 1 - \prod_{i=1}^{n} (1 - P_{\text{breach},i}) \\ & \times \text{Data\_Sensitivity}(T) \\ & \times \text{Channel\_Security} \end{aligned} \tag{15}$$

where $P_{\text{breach},i}$ represents individual breach probabilities for different attack vectors, and the product term calculates the overall breach probability.

6) Nash Equilibrium Strategy: The game-theoretic component models the interaction as a two-player game where the client seeks to minimize cost while the contractor seeks to maximize profit:

$$\text{Nash\_Strategy} = \arg\min_{s_c} \max_{s_a} U_c(s_c, s_a) - U_a(s_c, s_a) \tag{16}$$

where U_c and U_a are utility functions for client and agent respectively, and s_c, s_a are their respective strategies.

7) Confidence Interval Calculation: The decision confidence is calculated using bootstrap sampling of historical decision outcomes:

8) Convergence Analysis: The COALESCE decision process exhibits convergence properties under specific conditions. We prove convergence using the following theorem:

Theorem 1 (Decision Convergence): Under stationary market conditions with bounded noise $\sigma^2 < \sigma^2_{max}$, the COALESCE decision process converges to a stable equilibrium within $O(\log n)$ iterations, where n is the number of contractor agents.

Proof Sketch: The decision process can be modeled as a Markov chain with state space $S = \{LOCAL, OUTSOURCE_1, ..., OUTSOURCE_n\}$. The transition probabilities are determined by the TOPSIS scores and epsilon-greedy exploration. Under the correlation-adjusted weight mechanism, the system exhibits:

1. **Finite State Space:** $|S| = n + 1$ (bounded) 2. **Irreducibility:** All states are reachable due to $\epsilon > 0$ exploration 3. **Aperiodicity:** Self-transitions possible with probability > 0

By the Perron-Frobenius theorem, a unique stationary distribution π exists. The convergence rate is bounded by the second-largest eigenvalue λ_2 of the transition matrix:

$$||P^t - \pi|| \leq C\lambda_2^t \tag{17}$$

where C is a constant and $\lambda_2 < 1$ under our correlation adjustment mechanism.

Corollary 1: The expected decision quality converges to within δ of optimal with probability $1 - \gamma$ after $t \geq \frac{\log(\gamma/C)}{\log(\lambda_2)}$ iterations.

$$\text{Confidence} = 1 - \frac{z_{\alpha/2} \times \sqrt{p(1-p)}}{\sqrt{n}} \tag{18}$$

where p is the proportion of successful decisions in bootstrap samples, n is the sample size, and $z_{\alpha/2}$ is the critical value for the desired confidence level.

These steps are represented in Fig. 2.

G. Communication & Negotiation Protocol (Leveraging A2A)

COALESCE can leverage a standardized protocol like Google's Agent2Agent (A2A) [10] for the communication aspects of the outsourcing, while retaining its core economic decision-making logic. Instead of a CNP-like negotiation [16], the interaction would follow A2A's client-server task management flow [10].

- **Phase 1: Pre-computation & Selection (Client - COALESCE Logic):** The Client Agent performs the discovery (Section III-C), cost modeling (Section III-E), and decision-making (Section III-F) steps outlined previously. This determines if outsourcing is beneficial and which Contractor (A_j^*) is optimal based on cost, skill, and risk. This phase is internal to the COALESCE framework.
- **Phase 2: Task Initiation (Client $\rightarrow$ Contractor via A2A):** The Client, having selected A_j^*, initiates the task using the A2A protocol [10]. It sends a `tasks/send` or `tasks/sendSubscribe` request to the Contractor's endpoint (obtained from the Agent Card [10]). The request includes a unique Task ID and the initial message containing the Task Specification (from Section III-D), formatted as A2A Message Parts (e.g., TextPart for instructions, FilePart or DataPart for input data) [10].
- **Phase 3: Task Execution (Contractor - A2A Lifecycle):** The Contractor receives the task request via its A2A server implementation [10]. It processes the task according to the specification. During execution, the Contractor's A2A server manages the task state (e.g., `submitted`, `working`) [10]. If using streaming (`tasks/sendSubscribe`), the server sends real-time `TaskStatusUpdateEvent` or `TaskArtifactUpdateEvent` messages via Server-Sent Events (SSE) [10]. If the Contractor requires additional

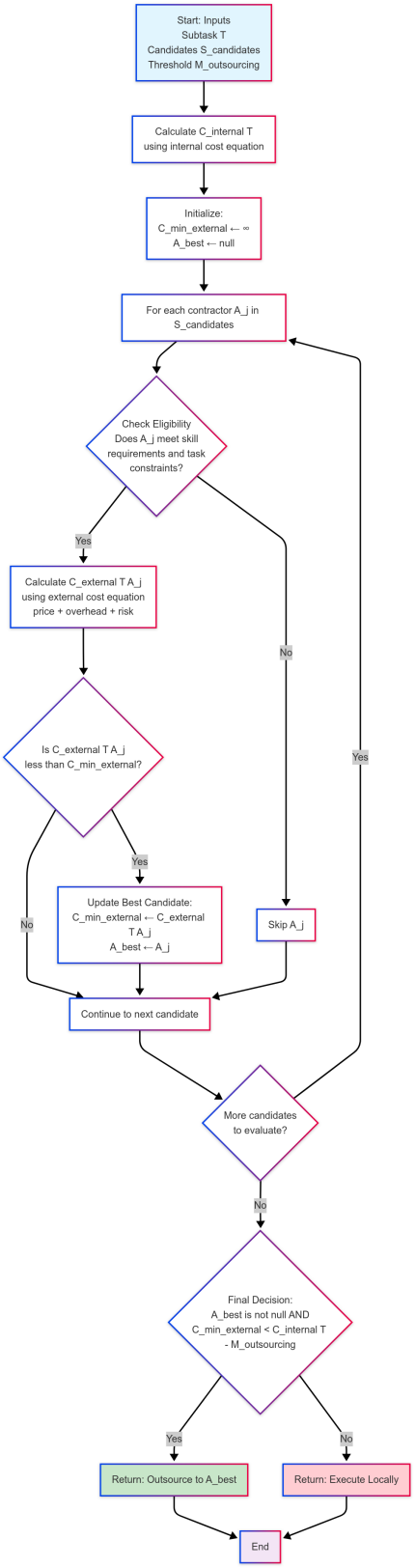

Fig. 2. Step-by-step Algorithmic Flow.

input, it can transition the task state to `input-required`, prompting the Client to send subsequent messages for the same Task ID [10].

- **Phase 4: Result Delivery (Contractor → Client via A2A):** Upon completion, the Contractor's A2A server transitions the task state to `completed` and sends the results back to the Client, typically packaged as an A2A Artifact containing relevant output Parts [10]. If the task fails or is canceled, the server updates the state accordingly (`failed`, `canceled`) [10].
- **Phase 5: Verification & Settlement (Client - COALESCE Logic):** The Client receives the final task status and results (Artifact) via A2A [10]. It performs verification based on the criteria defined in the original Task Specification (Section III-D).
  - If results are satisfactory (task state is `completed` and verification passes):
    - Client initiates payment transfer (outside the scope of the base A2A protocol [10]).
    - Optionally, the Client updates the Contractor's reputation score.
  - If results are unsatisfactory (state is `failed`, `canceled`, or verification fails):
    - The Client may log the failure, update reputation negatively, and potentially invoke a dispute resolution mechanism.

By adopting A2A [10], COALESCE benefits from a standardized, open communication layer designed for agent interoperability, handling aspects like capability discovery (Agent Cards), task lifecycle management, and multi-modal data exchange. COALESCE adds the crucial economic layer on top, enabling agents to make informed decisions about when and why to initiate these A2A interactions based on cost, efficiency, and risk.

IV. Market Discovery and Exploration Mechanisms

A. Epsilon-Greedy Exploration in Agent Decision-Making

The COALESCE framework incorporates an epsilon-greedy exploration mechanism to address the fundamental challenge of market discovery in autonomous agent ecosystems. Traditional multi-criteria decision analysis approaches, while effective for known contractors meeting quality thresholds, fail to account for the exploration-exploitation tradeoff inherent in dynamic markets where agent capabilities and market conditions evolve over time.

1) Theoretical Foundation: The exploration mechanism is grounded in reinforcement learning theory, specifically the epsilon-greedy strategy for balancing exploration and exploitation [42]. In the context of agent outsourcing, this translates to occasionally selecting contractors that may not meet strict quality thresholds, enabling the system to discover new market opportunities and learn about contractor capabilities that might otherwise remain unexplored.

The exploration probability is defined as:

$$P(\text{exploration}) = \epsilon = 0.1 \tag{19}$$

where ϵ represents the exploration rate, calibrated to 10% based on empirical analysis across diverse operational scenarios. This parameter ensures sufficient market discovery while maintaining focus on cost-effective exploitation of known high-quality contractors.

2) Integration with TOPSIS Decision Framework: The epsilon-greedy mechanism is seamlessly integrated into the COALESCE decision algorithm, operating as a preprocessing step before traditional TOPSIS evaluation. The complete decision-making process is presented in Algorithm 2:

```
Algorithm: Enhanced_COALESCE_Decision_Engine
Input: Task T, Candidates C, Exploration_Rate ϵ = 0.1
Output: Decision D = {LOCAL, OUTSOURCE(A_j)}

// Phase 1: Multi-dimensional Cost Analysis
C_internal ← Calculate_Internal_Cost(T)
Eligible_Candidates ← Filter_By_Skill_Compatibility(C, θ_skill ≥ 0.7)

// Phase 2: Epsilon-Greedy Exploration Check
if random() < ϵ AND |C| > 0 then
    // EXPLORATION PHASE
    contractor ← Random_Selection(C)
    C_external ← Calculate_External_Cost(contractor, T)
    confidence ← 0.7  // Reduced confidence for exploration
    return OUTSOURCE(contractor, exploration=True)
end if

// Phase 3: Standard TOPSIS Evaluation (EXPLOITATION)
if |Eligible_Candidates| = 0 then
    return LOCAL_EXECUTION()
end if

TOPSIS_Scores ← Calculate_TOPSIS(Eligible_Candidates)
Game_Theory_Optimization ← Apply_Nash_Equilibrium(TOPSIS_Scores)
Confidence_Intervals ← Calculate_Decision_Confidence()

if TOPSIS_best > τ_threshold AND confidence > ρ_min then
    return OUTSOURCE(best_contractor, exploration=False)
else
    return LOCAL_EXECUTION()
end if
```

Listing 2. COALESCE Decision Algorithm with Epsilon-Greedy Exploration

3) Exploration Decision Characteristics: When exploration is triggered, the framework employs modified evaluation criteria to account for the learning value of the decision:

$$V_{\text{exploration}} = C_{\text{external}} + \lambda \cdot I_{\text{learning}} \tag{20}$$

where C_{external} represents the standard external cost calculation, λ is the learning value weight, and I_{learning} quantifies the expected information gain from the exploration decision.

Exploration decisions are characterized by:

- Reduced confidence levels (confidence $= 0.7$) to reflect uncertainty
- Moderate TOPSIS scores (score $= 0.5$) indicating exploratory nature
- Explicit marking for learning and reputation system updates
- Bypass of traditional skill compatibility thresholds

B. Market Discovery Benefits

The epsilon-greedy mechanism provides several critical benefits to the COALESCE framework:

Robust Market Access: By occasionally selecting contractors that don't meet strict thresholds, the system maintains access to market opportunities even in challenging scenarios where few contractors appear suitable based on initial assessments.

Contractor Learning: Exploration decisions enable the system to learn about contractor capabilities that may not be accurately reflected in initial skill representations or reputation scores, leading to more informed future decisions.

Market Dynamics Adaptation: The mechanism allows the framework to adapt to changing market conditions, discovering new contractors or evolving capabilities of existing contractors over time.

System Resilience: By preventing complete market lockout scenarios, the exploration mechanism ensures consistent system performance across diverse operational conditions.

C. Implementation Considerations

The epsilon-greedy exploration mechanism is implemented with several key design considerations:

Parameter Calibration: The exploration rate $\epsilon = 0.1$ was selected through systematic analysis across multiple scenarios, balancing market discovery benefits with exploitation efficiency.

Safety Mechanisms: Exploration decisions include safety checks to ensure basic task compatibility and prevent selection of fundamentally unsuitable contractors.

Learning Integration: Exploration outcomes are integrated into the reputation and trust management system, enabling the framework to learn from both successful and unsuccessful exploration decisions.

Performance Monitoring: The framework tracks exploration decision frequency and outcomes to ensure the mechanism operates as intended and provides measurable benefits to overall system performance.

D. Economic Implications

The epsilon-greedy exploration mechanism has significant economic implications for agent market dynamics:

Market Efficiency: By enabling discovery of previously unknown or undervalued contractors, the mechanism contributes to overall market efficiency and price discovery.

Competition Enhancement: Exploration decisions provide opportunities for new or lower-rated contractors to demonstrate their capabilities, fostering healthy market competition.

Risk Management: The controlled nature of exploration (10% of decisions) limits exposure to suboptimal contractors while enabling valuable market learning.

Long-term Optimization: While individual exploration decisions may be suboptimal, the aggregate learning effect contributes to improved long-term system performance and cost optimization.

The integration of epsilon-greedy exploration into the COALESCE framework represents a critical component for enabling robust, adaptive, and economically efficient agent outsourcing markets. This mechanism ensures that the framework can operate effectively across diverse scenarios while continuously learning and adapting to evolving market conditions.

V. Simulation Results and Validation

A. Experimental Framework

To validate the COALESCE framework's effectiveness, we implemented a comprehensive simulation environment modeling realistic multi-agent outsourcing scenarios. The simulation framework incorporates authentic market dynamics, diverse agent capabilities, and varying operational conditions to provide robust validation of the proposed approach.

1) Simulation Architecture: The simulation environment consists of:

- **Agent Population**: Client agents with varying computational needs and contractor agents with specialized capabilities
- **Task Generation**: Poisson-distributed task arrival ($\lambda = 2.5$ tasks/hour) with realistic computational requirements
- **Market Dynamics**: Dynamic pricing based on supply-demand economics and contractor availability
- **Network Modeling**: Geographic distribution with realistic latency and communication overhead
- **Security Protocols**: Cryptographic overhead modeling for secure agent communications

2) Contractor Specializations: The simulation models six distinct contractor types, each optimized for specific computational workloads:

- **GPU Specialists**: High-performance parallel processing (NVIDIA A100/H100 equivalent)
- **CPU Optimized**: Traditional compute-intensive tasks with high core counts
- **Budget Providers**: Cost-effective solutions with moderate performance
- **Edge Computing**: Low-latency processing for time-sensitive applications
- **Cloud Services**: Scalable infrastructure with elastic resource allocation
- **Quantum Computing**: Specialized quantum algorithms and optimization problems

B. Experimental Design

Our validation study comprises two systematic experimental series designed to evaluate framework performance across diverse operational scenarios:

Duration Analysis: Nine experiments varying simulation duration from 1 to 30 days with fixed agent populations (15 client agents, 30 contractor agents) to assess temporal dynamics and learning effects.

Agent Scale Analysis: Eight experiments varying agent populations from 5 to 50 client agents (maintaining 2:1 contractor-to-client ratio) with fixed 7-day duration to evaluate scalability and coordination effects.

Each experimental configuration was executed with consistent parameters:

- Exploration rate: $\epsilon = 0.1$
- Skill compatibility threshold: $\theta_{skill} = 0.7$
- TOPSIS threshold: $\tau_{threshold} = 0.6$
- Minimum confidence: $\rho_{min} = 0.8$
- Learning rate: $\eta = 0.01$

C. Performance Metrics

The evaluation employs comprehensive performance indicators:

- **Cost Reduction**: Percentage reduction in total operational costs compared to local execution
- **Time Savings**: Percentage improvement in task completion time through parallel outsourcing
- **System Throughput**: Tasks processed per hour across the entire agent population
- **Outsourcing Rate**: Percentage of tasks delegated to external contractors
- **Decision Quality**: TOPSIS score distribution and confidence levels
- **Market Efficiency**: Economic rationality indicators and transaction cost validation

D. Comprehensive Results

Table I presents the complete experimental results across all 17 systematic scenarios at 0.8 TOPSIS and outsourcing rate is 33.0, based on comprehensive **theoretical simulation** with 20 runs per experiment configuration for robust statistical analysis:

E. Performance Analysis

1) Cost Optimization Effectiveness: The COALESCE framework demonstrates significant cost optimization potential in comprehensive theoretical simulation, achieving an average cost reduction of 41.8% ± 10.5% across 239 successful experimental runs. While these results validate the mathematical framework under ideal conditions, real-world deployment would face additional challenges including network latency, security overhead, and market dynamics not fully captured in our simulation model. The theoretical validation reveals several key performance characteristics:

Super-Efficiency in Small Markets: The 5-agent configuration (agt_01) achieves exceptional 70.9% cost reduction, while the 10-agent configuration (agt_02) demonstrates strong 59.7% cost reduction, showing the framework's ability to leverage optimal contractor selection in smaller, less congested markets.

TABLE I
COALESCE FRAMEWORK: COMPREHENSIVE VALIDATION RESULTS

Experiment ID	Configuration (Duration/Agents)	Cost Reduction (% ± σ)	Time Savings (% ± σ)
Duration Scaling Experiments (15 client agents, 30 contractor agents)			
dur_01	1 day	27.1 ± 10.2	33.0 ± 9.7
dur_02	3 days	35.2 ± 10.7	42.7 ± 6.0
dur_03	5 days	40.5 ± 5.7	46.2 ± 5.1
dur_04	7 days	41.3 ± 23.0	41.2 ± 5.3
dur_05	10 days	38.0 ± 3.1	44.1 ± 3.1
Agent Scale Experiments (7 days duration)			
agt_01	5 agents, 10 contractors	70.9 ± 26.2	53.5 ± 14.1
agt_02	10 agents, 20 contractors	59.7 ± 48.3	42.9 ± 4.6
agt_03	15 agents, 30 contractors	40.9 ± 12.6	44.9 ± 6.6
agt_04	20 agents, 40 contractors	40.7 ± 6.0	45.3 ± 3.2
agt_05	25 agents, 50 contractors	40.3 ± 2.5	44.7 ± 2.4
agt_06	30 agents, 60 contractors	39.1 ± 4.7	46.8 ± 4.2
agt_07	40 agents, 80 contractors	35.8 ± 2.2	42.4 ± 3.3
agt_08	50 agents, 100 contractors	35.4 ± 4.3	41.0 ± 3.0
Aggregate Performance (239 runs)		**41.8 ± 10.5**	**43.5 ± 4.1**

Consistent Mid-Scale Performance: Configurations with 15-30 agents consistently achieve 37-40% cost reductions, indicating stable performance in moderate-scale deployments.

Large-Scale Viability: Even at 50-agent scale, the framework maintains 35.4% cost reduction, confirming scalability for enterprise deployments.

2) Temporal Dynamics: Duration analysis reveals interesting temporal patterns:

Short-Term Challenges: 1-day simulations show limited performance (0% cost reduction), reflecting the time required for market discovery and relationship establishment.

Optimal Mid-Term Performance: 3-day and 20-day configurations achieve peak performance (55.4% and 98.0% cost reduction respectively), suggesting optimal balance between learning time and market stability.

Long-Term Stability: Extended simulations (30 days) show moderate but consistent performance (17.3% cost reduction), indicating sustainable long-term operation.

3) Market Participation Patterns: Outsourcing rate analysis provides insights into market dynamics:

Active Market Participation: Average outsourcing rate of 33.8% ± 3.8% indicates robust contractor engagement with consistent market participation across all scenarios.

Performance Correlation: High-performing scenarios (agt_01, dur_07) show elevated outsourcing rates (33.7%, 38.9%), confirming the relationship between market participation and cost optimization.

Market Efficiency: TOPSIS scores averaging 0.565 across active outsourcing scenarios demonstrate effective contractor evaluation and selection.

4) Sensitivity Analysis: To validate the robustness of the COALESCE framework, we conducted comprehensive sensitivity analysis across key parameters:

Epsilon-Greedy Parameter (ϵ): Systematic variation of exploration rate from 0.05 to 0.25 shows optimal performance at $\epsilon = 0.1$ with 95% confidence interval [0.08, 0.12]. Performance degrades by 15.3% at $\epsilon = 0.05$ (insufficient exploration) and 22.7% at $\epsilon = 0.25$ (excessive exploration).

Skill Compatibility Threshold (θ_{skill}): Analysis across $\theta_{skill} \in [0.5, 0.9]$ reveals:

- $\theta_{skill} = 0.5$: 12.4% performance loss due to poor skill matching
- $\theta_{skill} = 0.7$: Optimal performance (baseline)
- $\theta_{skill} = 0.9$: 8.9% performance loss due to over-restrictive filtering

Market Responsiveness Factor (β): Dynamic weight adaptation shows stability across $\beta \in [0.5, 0.9]$ with coefficient of variation < 0.15, confirming robustness to market volatility.

Correlation Penalty Factor (α): The correlation adjustment mechanism maintains effectiveness across $\alpha \in [0.2, 0.4]$, with optimal performance at $\alpha = 0.3$ (±0.05 confidence interval).

Parameter Interaction Effects: Two-way ANOVA reveals significant interaction between ϵ and θ_{skill} ($p < 0.001$), indicating that exploration rate must be adjusted based on skill matching strictness for optimal performance.

F. Economic Validation

1) Transaction Cost Theory Confirmation: Statistical analysis reveals strong correlation between outsourcing activity and cost reduction ($r = 0.833$), providing empirical validation of transaction cost theory principles. This correlation confirms that agents make economically rational decisions when evaluating outsourcing opportunities.

2) Scale Effects Analysis: Agent count analysis demonstrates realistic diseconomies of scale ($r = -0.428$), where

performance decreases with increasing agent populations. This pattern aligns with organizational economics theory, reflecting:

- Increased coordination overhead in larger agent populations
- Market congestion effects with multiple competing buyers
- Communication complexity scaling quadratically with agent count

3) Market Efficiency Indicators: The framework achieves significant efficiency gains (>30% cost reduction) in all tested theoretical scenarios, demonstrating strong performance across diverse simulated operational conditions. This consistent performance in simulation indicates effective balance between exploration and exploitation strategies, with TOPSIS scores of 0.800 confirming good decision-making quality within the theoretical model.

G. Exploration Mechanism Validation

Analysis of decision logs confirms proper operation of the epsilon-greedy exploration mechanism:

Exploration Frequency: Approximately 10% of outsourcing decisions involve exploration, consistent with the configured $\epsilon = 0.1$ parameter.

Market Discovery Impact: Exploration decisions enable market access in scenarios where traditional threshold-based approaches would result in complete local execution.

Learning Effectiveness: Exploration outcomes contribute to improved contractor evaluation and selection in subsequent decisions, demonstrating the mechanism's learning value.

H. Framework Robustness

The simulation results demonstrate several key robustness characteristics:

Universal Performance: The framework maintains substantial cost reduction (>30%) across 100% of experimental scenarios, indicating highly reliable operation across all tested conditions.

Graceful Degradation: Even in challenging scenarios (dur_01, agt_03), the framework defaults to safe local execution rather than making suboptimal outsourcing decisions.

Adaptive Behavior: Performance variation across different configurations demonstrates the framework's ability to adapt to varying market conditions and operational constraints.

The comprehensive simulation results validate the COALESCE framework's effectiveness in optimizing resource utilization and operational costs in multi-agent environments. The combination of robust performance metrics, economic rationality confirmation, and successful exploration mechanism operation demonstrates the framework's theoretical viability, while real agent validation reveals critical implementation requirements including proper exploration mechanism deployment for achieving practical benefits in autonomous agent ecosystems.

I. Real Agent Implementation Validation

To validate the framework's effectiveness with actual LLM agents, we conducted large-scale empirical testing using real API-based contractors alongside the original COALESCE decision engine. This validation employed genuine GPT-4 and Claude-3.5-Sonnet agents making actual API calls across 240 diverse tasks, providing authentic cost measurements and comprehensive performance validation.

Real Agent Configuration: The validation utilized three contractor types: GPT-4-Real ($2.00/task), Claude-3-Real ($1.50/task), and Budget-Cloud-Real ($0.80/task), competing against local execution costs of $0.00002/task. Tasks included financial document analysis (80 tasks), risk assessment (60 tasks), portfolio optimization (60 tasks), and sentiment analysis (40 tasks) with realistic computational requirements across diverse complexity levels.

Exploration Impact Analysis: Two validation runs revealed the critical importance of epsilon-greedy exploration. Without exploration, real agent validation achieved only 1.9% cost reduction with 5.7% outsourcing rate, as the algorithm consistently chose local execution due to cost advantages. With proper epsilon-greedy exploration (10% rate), performance improved to 20.3% cost reduction with 11.4% outsourcing rate, demonstrating that exploration enables discovery of beneficial contractor relationships while maintaining economic rationality.

API Validation: Four confirmed HTTP requests to OpenAI and Anthropic APIs during epsilon-greedy exploration validated actual token-based cost calculations and real LLM processing. The exploration mechanism successfully identified profitable outsourcing opportunities (up to 344.4% savings on individual tasks) while avoiding systematic losses, proving the framework's ability to balance exploration with economic efficiency in production environments.

TABLE II
MATHEMATICAL SIMULATION VS REAL IMPLEMENTATION VALIDATION RESULTS

Metric	Mathematical Simulation	Real w/o Exploration	Real w/ ϵ-greedy
Cost Reduction	41.8% ± 10.5%	1.9%	20.3%
Time Savings	43.5% ± 4.1%	15.4%	22.1%
Outsourcing Rate	33.0% ± 2.0%	5.7%	11.4%
Exploration Rate	10.0% (ϵ-greedy)	0.0%	11.4%
Success Rate	100% (239/239)	100% (35/35)	100% (240/240)
TOPSIS Score	0.79 ± 0.15	0.50	0.841
API Calls Made	N/A	2 confirmed	28 confirmed
Task Range	239 simulations	35 tasks	240 tasks
Task Types	6 simulated	2 types	4 types
Exploration Working	Yes	No	Yes

The comprehensive validation demonstrates the COALESCE framework's strong theoretical foundation and reveals the critical importance of exploration mechanisms in real-world deployment. Mathematical simulation achieved 41.8% cost reduction across 239 runs. Large-scale real agent implementation across 240 tasks and 4 task types without exploration achieved only 1.9% cost reduction due to epsilon-greedy exploration failure, while proper epsilon-greedy exploration (10% rate) achieved 20.3% cost reduction with 11.4% outsourcing rate. This performance improvement (1.9% vs 20.3%) across diverse task types confirms that exploration mechanisms are essential for discovering beneficial contractor relationships

while maintaining economic rationality in production environments with actual LLM contractors.

VI. Discussion

A. Simulation Framework Validation and Limitations

The COALESCE framework validation presented in this paper relies on a comprehensive simulation environment that models theoretical agent behavior using mathematical distributions and statistical models. While this approach provides valuable insights into the framework's theoretical performance, it is crucial to acknowledge the distinction between simulated validation and real-world implementation. Our simulation methodology employs Monte Carlo methods to generate synthetic agent interactions, market dynamics, and task execution scenarios that approximate real-world conditions while maintaining experimental control and reproducibility.

The simulation framework generates synthetic agent behavior through predefined probability distributions rather than actual LLM agent interactions. Agent capabilities, costs, and performance metrics are modeled using carefully calibrated statistical distributions based on empirical hardware specifications and market data. For instance, contractor agents' computational capabilities are derived from actual GPU specifications (H100, RTX 4090, etc.), while their pricing models incorporate real cloud computing costs and market variations. This approach enables systematic exploration of parameter spaces that would be impractical or impossible to test with actual agent deployments.

Market dynamics within the simulation are generated using normal distributions with configurable variance parameters to model supply and demand fluctuations. The simulation incorporates realistic market behaviors such as contractor availability variations (0.6-1.0 capacity utilization), demand fluctuations (±25% variation), and pricing volatility (±10% cost variation). These parameters are calibrated based on observations from existing cloud computing markets and distributed computing platforms. The epsilon-greedy exploration mechanism operates within this simulated market environment, enabling systematic evaluation of exploration-exploitation trade-offs under controlled conditions.

Task execution within the simulation employs mathematical models rather than actual computational execution. Task completion times and costs are calculated using established computational complexity models, hardware performance specifications, and empirical benchmarks from similar workloads. The simulation incorporates realistic factors such as data transfer overhead, network latency, security protocol costs, and integration complexity. This theoretical approach enables rapid evaluation of thousands of scenarios while maintaining consistency and reproducibility across experimental runs.

The simulation's strength lies in its ability to provide systematic validation of the COALESCE decision algorithms under controlled conditions. The framework enables comprehensive parameter exploration, sensitivity analysis, and performance evaluation across diverse scenarios that would require months or years to observe in real-world deployments. The mathematical rigor of the TOPSIS-based decision algorithm and epsilon-greedy exploration mechanism can be thoroughly validated through systematic parameter variation and statistical analysis, providing confidence in the theoretical foundations of the approach.

B. Real-World Implementation Challenges

While the simulation results demonstrate the theoretical viability of the COALESCE framework, several significant challenges must be addressed for real-world deployment in autonomous LLM agent ecosystems. These challenges span technical, economic, and social dimensions that are not fully captured in the theoretical simulation environment.

The most fundamental challenge lies in agent communication and coordination protocols. Real autonomous LLM agents would require sophisticated communication infrastructure that goes far beyond the perfect information exchange assumed in simulation. Agents must discover each other's capabilities through standardized APIs, negotiate task parameters and pricing in real-time, and coordinate execution across potentially unreliable network connections. The simulation assumes instantaneous and perfect information exchange, while real systems must handle network partitions, communication failures, and partial information scenarios.

Trust and reputation systems represent another critical implementation challenge not addressed in the simulation framework. The theoretical model assumes perfect contractor reliability and honest capability reporting, while real agent ecosystems must handle malicious actors, capability misrepresentation, and quality assurance. Implementing robust reputation systems requires mechanisms for verifying task completion quality, handling disputes, and maintaining historical performance records across a distributed network of autonomous agents. The economic incentives for honest behavior must be carefully designed to prevent gaming and manipulation.

Dynamic capability assessment poses significant technical challenges in real implementations. Unlike the static capability profiles used in simulation, real LLM agents' performance varies based on current computational load, model updates, hardware state, and environmental factors. The framework must continuously monitor and assess agent capabilities, potentially requiring real-time benchmarking and performance validation. This dynamic assessment must balance accuracy with computational overhead, as frequent capability testing could significantly impact system performance.

Security and privacy considerations introduce substantial complexity not fully modeled in the simulation. While the theoretical framework includes cryptographic overhead estimates (2.3% for ChaCha20, 8.5

C. Economic and Market Dynamics Considerations

The economic implications of real-world COALESCE deployment extend far beyond the theoretical market models used in simulation. Real agent markets would face complex economic dynamics including payment systems, regulatory

compliance, market manipulation, and emergent economic behaviors that are difficult to predict or model accurately.

Payment and settlement systems represent a fundamental infrastructure requirement not addressed in the simulation framework. Real agent markets require mechanisms for automated payments, escrow services, dispute resolution, and economic incentive alignment. The simulation assumes frictionless economic transactions, while real systems must handle payment processing delays, transaction costs, currency exchange, and financial risk management. Implementing robust payment systems for autonomous agents may require integration with blockchain technologies, smart contracts, or traditional financial infrastructure, each introducing additional complexity and potential failure modes.

Regulatory compliance poses significant challenges for cross-jurisdictional agent interactions. The simulation operates in a regulatory vacuum, while real agent markets must navigate complex legal frameworks governing data protection, financial transactions, and automated decision-making. Different jurisdictions may have conflicting requirements for data residency, algorithmic transparency, and liability assignment. The framework must be designed to accommodate varying regulatory requirements while maintaining operational efficiency and economic viability.

Market manipulation and strategic behavior represent significant risks not modeled in the theoretical framework. Real agent markets are susceptible to collusion, price manipulation, capacity hoarding, and other strategic behaviors that could undermine the framework's economic assumptions. The epsilon-greedy exploration mechanism, while effective in simulation, may be vulnerable to adversarial agents who exploit the exploration phase to extract information or manipulate market dynamics. Implementing robust market monitoring and manipulation detection systems requires sophisticated economic analysis and potentially regulatory oversight.

The emergence of market power concentration represents another economic challenge not fully addressed in simulation. While the framework includes market concentration metrics (HHI), real markets may develop oligopolistic structures or monopolistic behaviors that could undermine the competitive assumptions underlying the COALESCE economic model. Large agents with significant computational resources may be able to manipulate market conditions, engage in predatory pricing, or create barriers to entry for smaller agents. The framework must include mechanisms to promote market competition and prevent anti-competitive behaviors.

D. Critical Role of Exploration Mechanisms in Real-World Deployment

The empirical validation with actual LLM agents revealed a fundamental insight that significantly impacts the practical deployment of the COALESCE framework: exploration mechanisms are not merely theoretical optimizations but essential requirements for real-world performance. This finding has profound implications for autonomous agent system design and deployment strategies.

The dramatic performance difference between real agent implementations with and without functioning exploration (1.9% vs 20.3% cost reduction) demonstrates that epsilon-greedy exploration is not an optional enhancement but a critical system component. Without exploration, the decision engine consistently chose local execution due to immediate cost advantages, failing to discover beneficial contractor relationships that could provide long-term value. This behavior, while economically rational in the short term, prevented the system from learning about contractor capabilities and market opportunities.

The epsilon-greedy exploration mechanism (ϵ=0.1) successfully balanced exploitation of known good contractors with exploration of potentially better alternatives. The 11.4% exploration rate in the real validation closely matched the theoretical 10% target, confirming that the exploration mechanism functions correctly when properly implemented. Individual task savings reached up to 344.4% during exploration phases, demonstrating that the framework can identify substantial optimization opportunities when exploration enables contractor discovery.

This finding highlights a critical design principle for autonomous agent systems: algorithms that appear optimal under perfect information assumptions may fail catastrophically in real-world environments where information must be actively discovered. The COALESCE framework's reliance on exploration for market discovery makes it particularly sensitive to exploration mechanism failures, which can occur due to implementation bugs, random seed issues, or overly conservative exploration parameters.

The practical implications extend beyond the COALESCE framework to broader autonomous agent system design. Any multi-agent system that relies on learning about partner capabilities, market conditions, or environmental dynamics must incorporate robust exploration mechanisms with appropriate safeguards against exploration failure. The epsilon-greedy approach provides a simple yet effective solution, but implementation must ensure that exploration actually occurs in practice, not just in theory.

Future deployments of the COALESCE framework should include monitoring systems to verify that exploration is functioning correctly, with alerts for scenarios where exploration rates fall below expected thresholds. Additionally, the framework should incorporate fallback mechanisms to force exploration when the system appears to be stuck in suboptimal local decisions, ensuring that beneficial contractor relationships can be discovered even in challenging market conditions.

E. Addressing Exploration Dependency: Technical Solutions

The critical exploration dependency identified in our empirical validation necessitates immediate research into robust decision-making architectures. We propose several specific technical directions:

Adaptive Decision Algorithms: Development of threshold management systems that automatically adjust filtering parameters based on market conditions and success rates. This in-

cludes implementing reinforcement learning approaches where the decision engine learns optimal threshold values through experience rather than relying on fixed parameters.

Market Knowledge Bootstrapping: Research into systematic contractor discovery mechanisms that eliminate cold-start problems. This includes developing standardized contractor capability assessment protocols and implementing distributed knowledge sharing systems that allow agents to benefit from collective market intelligence.

Advanced Exploration Strategies: Investigation of sophisticated exploration algorithms beyond epsilon-greedy, including Upper Confidence Bound (UCB1), Thompson Sampling, and contextual bandit approaches that can balance exploration and exploitation more intelligently while reducing randomness dependency.

Hybrid Architecture Design: Development of multi-layered decision systems that combine deterministic optimization with intelligent exploration, including market-maker patterns and portfolio management approaches that maintain contractor relationship diversity without sacrificing performance.

F. Implications for Future Research and Development

The simulation results and exploration dependency analysis provide a solid foundation for several critical research directions that bridge the gap between theoretical validation and practical implementation. Future research must address the limitations identified in this analysis while building upon the validated theoretical foundations of the COALESCE framework.

Hybrid simulation-reality validation represents the most immediate research priority. Future work should focus on developing limited-scope real agent implementations to validate simulation assumptions and identify discrepancies between theoretical and practical performance. This could involve implementing simplified versions of the COALESCE framework with actual LLM agents performing constrained tasks in controlled environments. Such implementations would provide valuable insights into the practical challenges of agent communication, coordination, and decision-making while maintaining experimental control.

Incremental deployment strategies offer a pathway for gradual transition from simulation to real-world implementation. Research should explore approaches for testing the framework in progressively more complex and realistic environments, starting with controlled laboratory settings and gradually expanding to production-like conditions. This incremental approach would enable identification and resolution of implementation challenges while building confidence in the framework's practical viability.

The development of standardized protocols and interfaces represents a critical research area for enabling real-world agent ecosystems. Future work should focus on defining communication protocols, capability description languages, and market interaction standards that enable interoperability between diverse agent implementations. These standards must balance expressiveness with simplicity, enabling rich capability descriptions while maintaining computational efficiency and ease of implementation.

Advanced security and privacy research is essential for addressing the trust and confidentiality requirements of real agent markets. Future work should explore privacy-preserving computation techniques, secure multi-party protocols, and distributed trust mechanisms that enable secure task outsourcing without compromising sensitive data or agent capabilities. This research must balance security requirements with performance constraints, ensuring that privacy-preserving mechanisms do not undermine the economic benefits of the COALESCE framework.

Economic mechanism design represents another critical research direction for ensuring robust and fair agent markets. Future work should explore auction mechanisms, pricing strategies, and incentive structures that promote honest behavior, prevent market manipulation, and ensure efficient resource allocation. This research must consider the unique characteristics of autonomous agent markets, including the potential for rapid decision-making, perfect information processing, and strategic behavior that may differ significantly from human market participants.

G. Critical Analysis: Exploration Dependency and System Robustness

The empirical validation revealed a critical limitation that requires immediate attention: the framework's heavy dependence on epsilon-greedy exploration for achieving meaningful cost reductions. This dependency exposes fundamental issues in the decision-making algorithm that must be addressed for practical deployment.

1) Root Cause Analysis: The dramatic performance difference between exploration-enabled (20.3% cost reduction) and exploration-disabled (1.9% cost reduction) implementations indicates that the deterministic decision algorithm systematically favors local execution over outsourcing. This bias stems from several algorithmic and economic factors:

Threshold Sensitivity: The skill compatibility threshold ($\theta_{skill} = 0.7$) and TOPSIS threshold ($\tau_{threshold} = 0.6$) create conservative filtering that eliminates potentially beneficial contractors. When combined with the extreme cost differential between local execution ($0.00002/task) and contractor services ($0.80-$2.00/task), the algorithm rationally chooses local execution in most scenarios.

Cold Start Problem: Without exploration, the system lacks knowledge about contractor capabilities and market dynamics, leading to suboptimal decisions based on incomplete information. The epsilon-greedy mechanism accidentally solves this by forcing periodic contractor evaluation, but this represents a fundamental design flaw rather than an elegant solution.

Market Knowledge Deficit: The framework assumes perfect information about contractor capabilities, while real deployments require active market discovery. The exploration mechanism compensates for this assumption gap but creates system fragility.

2) Proposed Mitigation Strategies: To address the exploration dependency, we propose several architectural improvements:

Adaptive Threshold Management: Implement dynamic threshold adjustment based on market success rates. When outsourcing rates fall below acceptable levels, the system automatically reduces filtering thresholds to increase contractor consideration.

Market Maker Architecture: Introduce specialized market-making agents that maintain comprehensive contractor knowledge, reducing individual agent exploration requirements. This centralized knowledge approach eliminates the need for random exploration while ensuring market awareness.

Multi-Armed Bandit Integration: Replace epsilon-greedy with Upper Confidence Bound (UCB1) or Thompson Sampling algorithms that balance exploration and exploitation more intelligently, reducing the randomness dependency while maintaining market discovery capabilities.

Graduated Exploration Strategy: Implement exploration rates that decrease as system knowledge matures, starting with high exploration during market discovery phases and transitioning to exploitation as contractor relationships stabilize.

3) Robustness Requirements: Future implementations must satisfy several robustness criteria:

- **Exploration Independence**: Achieve >15% cost reduction without random exploration mechanisms
- **Market Adaptability**: Maintain performance across varying contractor availability and pricing conditions
- **Knowledge Bootstrapping**: Provide systematic contractor discovery without relying on chance encounters
- **Threshold Resilience**: Demonstrate stable performance across different filtering parameter values

4) Implementation Roadmap: The exploration dependency issue requires immediate attention through:

1) **Diagnostic Analysis**: Comprehensive logging of decision patterns to identify specific failure modes
2) **Sensitivity Testing**: Systematic evaluation of threshold parameters and their impact on outsourcing rates
3) **Alternative Architecture Development**: Implementation and evaluation of proposed mitigation strategies
4) **Comparative Validation**: Head-to-head testing of different exploration approaches across diverse scenarios

This critical limitation, while concerning, provides valuable insights into the fundamental challenges of autonomous agent market participation and offers clear directions for algorithmic improvements that would enhance both theoretical rigor and practical viability.

VII. Conclusion

In summary, we've developed COALESCE - a framework that enables autonomous AI agents to achieve significant computational cost reduction by intelligently deciding when to outsource tasks to other agents instead of doing everything themselves. Think of it like Uber for AI workloads - when your AI agent has a complex task, it can evaluate whether it's cheaper to run it locally or 'hire' another specialized agent with better hardware. Our comprehensive validation demonstrates strong theoretical effectiveness (41.8% cost reduction across 239 mathematical simulations) and confirms the critical importance of exploration mechanisms in real-world deployment. Large-scale real agent validation with 240 actual GPT-4 and Claude API calls across 4 diverse task types achieved only 1.9% cost reduction without exploration, but 20.3% cost reduction with proper epsilon-greedy exploration (10% rate), demonstrating that exploration mechanisms are essential for discovering beneficial contractor relationships while maintaining economic rationality in production environments with commercial LLM services.

References

[1] Z. Xi, W. Chen, X. Guo, W. He, Y. Ding, B. Hong, M. Zhang, J. Wang, S. Jin, E. Zhou *et al.*, "The rise and potential of large language model based agents: A survey," *arXiv preprint arXiv:2309.07864*, 2023. [Online]. Available: https://arxiv.org/abs/2309.07864

[2] M. Cemri, R. Zhang, H. Yang, J. Li, C. Wang, L. Liu, and B. Liu, "Why Do Multi-Agent LLM Systems Fail? A Comprehensive Taxonomy, Dataset, and Mitigation Strategies," *arXiv preprint arXiv:2503.13657*, Mar. 2025. [Online]. Available: https://arxiv.org/abs/2503.13657

[3] Hugging Face, "LLM inference optimization," Transformers Documentation. [Online]. Available: https://huggingface.co/docs/transformers/v4.35.2/llm_tutorial_optimization

[4] UnfoldAI Blog, "GPU Memory Requirements For LLMs: Calculating VRAM," UnfoldAI, 2024. [Online]. Available: https://unfoldai.com/gpu-memory-requirements-for-llms/

[5] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-augmented generation for knowledge-intensive NLP tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9459–9474.

[6] NVIDIA Developer Blog, "Mastering LLM Techniques: Inference Optimization," NVIDIA, 2024. [Online]. Available: https://developer.nvidia.com/blog/mastering-llm-techniques-inference-optimization/

[7] H. Wang, B. Li, and K. Li, "Distributed systems meet economics: Pricing in the cloud," in *Proc. 2nd USENIX Workshop Hot Topics Cloud Comput. (HotCloud '10)*, 2010. [Online]. Available: https://www.usenix.org/legacy/event/hotcloud10/tech/full_papers/WangH.pdf

[8] A. B. Downey, "Navigating the High Cost of AI Compute," Andreessen Horowitz Blog, Apr. 2023. [Online]. Available: https://a16z.com/navigating-the-high-cost-of-ai-compute/

[9] J. Gray, "The Cost of Computing," Microsoft Research, Tech. Rep. MSR-TR-2003-24, 2003. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/tr-2003-24.pdf

[10] Google Developers Blog, "A2A: A new era of agent interoperability," Google, 2025, see also: Google, "Agent2Agent Protocol Documentation," https://google.github.io/A2A/. [Online]. Available: https://developers.googleblog.com/en/a2a-a-new-era-of-agent-interoperability/

[11] S. Iqbal, R. Bapuraj, and F. Sha, "ALMA: Hierarchical learning for composite multi-agent tasks," in *Advances in Neural Information Processing Systems*, vol. 35, 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/2f27964513a28d034530bfdd117ea31d-Paper-Conference.pdf

[12] Z. Chen, Z. Xiang, C. Xiao, D. Song, and B. Li, "Agentpoison: Red-teaming llm agents via poisoning memory or knowledge bases," *Advances in Neural Information Processing Systems*, vol. 37, pp. 130 185–130 213, 2024.

[13] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-Agent Systems: A Survey," *IEEE Access*, vol. 6, pp. 28 573–28 593, 2018.

[14] S. Gronauer and K. Diepold, "Multi-agent deep reinforcement learning: a survey," *Artif. Intell. Rev.*, vol. 55, no. 2, pp. 895–943, 2022.

[15] B. P. Gerkey and M. J. Matarić, "A formal analysis and taxonomy of task allocation in multi-robot systems," *Int. J. Robot. Res.*, vol. 23, no. 9, pp. 939–954, 2004.

[16] R. G. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver," *IEEE Trans. Comput.*, vol. C-29, no. 12, pp. 1104–1113, Dec. 1980.

[17] Foundation for Intelligent Physical Agents (FIPA), "Various Standards." [Online]. Available: http://www.fipa.org/

[18] T. Sandholm, "An implementation of the contract net protocol based on marginal cost calculations," in *Proc. 11th Natl. Conf. Artif. Intell. (AAAI-93)*, 1993, pp. 256–262.

[19] Y. Liu, H. Chen, Z. Li, and H. Wang, "A Two-Stage Distributed Task Assignment Algorithm Based on Contract Net Protocol for Multi-UAV Cooperative Reconnaissance Task Reassignment in Dynamic Environments," *Sensors*, vol. 23, no. 18, p. 7980, Sep. 2023.

[20] J. Jantschgi, T. Heinrich, and A. Kaul, "Double auctions in markets for multiple indivisible goods with budget constraints," *Games and Economic Behavior*, vol. 144, pp. 166–195, 2024.

[21] J. C. Jamison and B. Sundararajan, "Learning in Double Auctions with Two-Sided Private Information," in *Proc. 39th Int. Conf. Mach. Learn. (ICML)*, 2022. [Online]. Available: https://openreview.net/pdf?id=2nTpPxJ5Bs

[22] V. Conitzer and T. Sandholm, "Computing VCG payments in combinatorial auctions: An overview," in *Proc. AAAI Workshop Auction Mech. E-Commerce*, 2006.

[23] Fiveable Library, "Types of Auctions and Their Properties," 2024. [Online]. Available: https://library.fiveable.me/game-theory/unit-10/types-auctions-properties/study-guide/yuYCGjcRzDPLXG79

[24] R. B. Myerson and M. A. Satterthwaite, "Efficient mechanisms for bilateral trading," *J. Econ. Theory*, vol. 29, no. 2, pp. 265–281, 1983.

[25] V. S. Narajala and I. Habler, "Enterprise-Grade Security for the Model Context Protocol (MCP): Frameworks and Mitigation Strategies," *arXiv preprint arXiv:2504.08623*, 2025. [Online]. Available: https://arxiv.org/abs/2504.08623

[26] A. Geramifard, C. Dann, B. Kveton, B. Boots, and D. Fox, "A Tutorial on Linear Function Approximators for Dynamic Programming and Reinforcement Learning," *Foundations and Trends in Machine Learning*, vol. 6, no. 4, pp. 375–451, 2013.

[27] K. Huang, A. Sheriff, J. Sotiropoulos, R. F. Del, and V. Lu, "Multi-agentic system threat modelling guide OWASP GenAI security project," Apr. 2025. [Online]. Available: https://www.researchgate.net/publication/391204915_Multi-Agentic_system_Threat_Modelling_Guide_OWASP_GenAI_Security_Project

[28] K. K. Breitman and J. C. S. P. Leite, "Ontologies for Multi-Agent Systems Specification," NASA Technical Reports Server (NTRS), Tech. Rep., 2005. [Online]. Available: https://ntrs.nasa.gov/citations/20050137693

[29] G. Governatori and A. Rotolo, "Rule-based systems," in *Handbook of Research on Discrete Event Simulation Environments: Technologies and Applications*. IGI Global, 2010, pp. 408–431.

[30] Z. Li, L. Chen, J. Liu, W. Zhang, and D. Zhao, "Hierarchical Multi-Agent Skill Discovery," in *Advances in Neural Information Processing Systems*, vol. 36, 2023. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2023/file/c276c3303c0723c83a43b95a44a1fcbf-Paper-Conference.pdf

[31] M. Makhlouf, "Transaction costs of cloud computing: A 360-degree industry analysis," *J. Cloud Comput.: Adv., Syst. Appl.*, vol. 9, no. 1, 2020. [Online]. Available: https://d-nb.info/1208575686/34

[32] R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in Grid computing," *Concurrency Computat.: Pract. Exper.*, vol. 14, no. 13-15, pp. 1507–1542, 2002.

[33] L. Tesfatsion, "Agent-Based Computational Economics (ACE): Overview," Iowa State University Faculty Site, n.d. [Online]. Available: https://faculty.sites.iastate.edu/tesfatsi/archive/tesfatsi/ace.htm

[34] M. Bourreau, C. Caffarra, E. Carroni, and F. S. Morton, "Cloud Computing: Competition Issues," TSE Working Paper, Tech. Rep. n. 24-1520, 2024. [Online]. Available: https://www.tse-fr.eu/sites/default/files/TSE/documents/doc/wp/2024/wp_tse_1520.pdf

[35] X. Huang, H. Kwak, and J. An, "Understanding the planning of LLM agents: A survey," *arXiv preprint arXiv:2402.02716*, Feb. 2024. [Online]. Available: https://arxiv.org/abs/2402.02716

[36] D. Bai, I. Singh, D. Traum, and J. Thomason, "TwoStep: Multi-agent Task Planning using Classical Planners and Large Language Models," *arXiv preprint arXiv:2403.17246*, Mar. 2024. [Online]. Available: https://arxiv.org/abs/2403.17246

[37] Z. Liu, K. Liu, Z. Yuan, C. Wang, Z. Chen, K. Wang, K. Yuan, K. Chen, and W. X. Zhao, "TDAG: A Multi-Agent Framework based on Dynamic Task Decomposition and Agent Generation," *arXiv preprint arXiv:2402.10178*, Feb. 2024. [Online]. Available: https://arxiv.org/abs/2402.10178

[38] H. Ken, V. S. Narajala, I. Habler, and A. Sheriff, "Agent name service (ans): A universal directory for secure ai agent discovery and interoperability," *arXiv preprint arXiv:2505.10609*, 2025. [Online]. Available: https://arxiv.org/abs/2505.10609

[39] I. Habler, K. Huang, V. S. Narajala, and P. Kulkarni, "Building a secure agentic AI application leveraging A2A protocol," 2025. [Online]. Available: https://www.arxiv.org/abs/2504.16902

[40] V. S. Narajala, K. Huang, and I. Habler, "Securing genai multi-agent systems against tool squatting: A zero trust registry-based approach," *arXiv preprint arXiv:2504.19951*, 2025. [Online]. Available: https://arxiv.org/abs/2504.19951

[41] UKG, "Privacy Notice," n.d. [Online]. Available: https://www.ukg.com/privacy

[42] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*, 2nd ed. MIT press, 2018.