

Centralized Orchestration for High-Frequency Agent-to-Agent Negotiations: The Puppeteer Pattern

Aditya Rautaray 

Authors Information:

Cloud Security Service, CVS Healthcare,
USA

Correspondence:

aditya.rautaray@cvshealth.com

Article History:

Date Received: May 17, 2026

Date Revised: May 26, 2026

Date Accepted: June 1, 2026

Date Published: June 20, 2026

DOI:

<https://doi.org/10.5281/zenodo.20763208>

Document ID: 2026IJRMS0108

Recommended Citation:

Rautaray, A. (2026). Centralized Orchestration for High-Frequency Agent-to-Agent Negotiations: The Puppeteer Pattern. *International Journal of Research on Multidisciplinary Studies*, 1(5), 135–152. <https://doi.org/10.5281/zenodo.20763208>

Abstract. As AI agents become more common in complex systems, coordinating how they negotiate with each other in fast-moving environments has become a real challenge. The older, decentralized approaches — where each agent largely fend for itself — tend to fall apart at scale. They're slow, they often produce outcomes that are good for one agent but bad for the overall system, and they can't keep up when conditions are changing rapidly and every millisecond counts. This paper presents the Puppeteer Pattern, a new architecture that puts a central coordinator in charge of guiding negotiation strategies across a mix of different AI agents. That central controller uses machine learning to update strategies on the fly, manage shared resources, and keep the system secure all in real time. The results from testing are encouraging. The system achieved a negotiation success rate of 92.3%, delivered nearly 29% better outcomes for the group as a whole compared to fully decentralized setups, and responded in an average of 150 milliseconds compared to 920 milliseconds without central coordination. That's roughly six times faster. The paper does not only talk about the good results. It also honestly explains what you actually need to make this system work in the real world, things like the right technology setup, how to keep it safe from threats, whether it works with different kinds of AI agents, and what its weaknesses are so that developers can be prepared.

Keywords. Agentic AI; Multi-Agent Systems; Centralized Control; High-Frequency Negotiation; Cybersecurity

Introduction

Autonomous agents and multi-agent systems constitute foundational elements in contemporary computing, extending from distributed systems engineering to the integration of AI subdisciplines (Jennings et al., 2001). Intelligent agents—ranging from software programs to physical robots—increasingly participate in intricate scenarios characterized by varied capabilities, preferences, and competing objectives (An, 2022). As Agentic Artificial Intelligence (AAI) continues to evolve, these systems now operate in environments that are increasingly complex, unpredictable, and large-scale, driving the need for better coordination architectures (Adabara et al., 2025).

Recent advances in multi-agent reinforcement learning and distributed AI underscore the need for architectures that balance autonomy and centralized supervision to optimize collective outcomes (Xu & Zhao, 2020). Automated negotiation—critical for multi-agent resource allocation—is gaining particular attention in e-commerce, cloud resource management, energy trading, and microgrid decision support (An, 2022; Xu et al., 2017; Khan et al., 2025; Pinto et al., 2018).

Despite advances in distributed negotiation protocols, persistent challenges remain: coordination burden, inconsistent enforcement, and adversarial risks in fast-paced, heterogeneous agent environments. Without a unified strategy layer, negotiations are often suboptimal, systems are fragile, and security threats are amplified.

High-frequency A2A negotiation settings introduce additional complexity. Uncertain information, many counterparties, and self-interested agents make it difficult to find globally balanced strategies (An, 2022). While decentralized protocols grant agents autonomy, they frequently produce suboptimal collective outcomes, increased communication overhead, and elevated adversarial risk—especially where rapid responses are required (Khan et al., 2025; Day et al., 2017). Emerging security and governance challenges in AAI systems further demand stronger oversight than current decentralized methods provide (Suggu, 2025; Adabara et al., 2025).

Research Questions

This study is guided by the following research questions:

1. Does a centralized AI/ML-driven orchestration layer (the Puppeteer Pattern) yield statistically significant improvements in Negotiation Success Rate (NSR), social welfare, and resource allocation efficiency compared to decentralized and partially centralized multi-agent negotiation paradigms in high-frequency A2A environments?
2. What is the relative contribution of each architectural component (centralized control, AI adaptation, cybersecurity module, and feedback loop) to the overall performance of the Puppeteer Pattern?
3. Under what infrastructure and security prerequisites can the Puppeteer Pattern be deployed in real-world production environments, and what are its scalability boundaries?

Scope and Delimitations

This study focuses on high-frequency A2A negotiations in cloud-based multi-agent environments where negotiation round-trip times must remain below 200 ms. The evaluation is bounded to simulated synthetic agent populations of up to 1,000 agents using the Alternating Offers Protocol (AOP) with three negotiation issues ($K = 3$). The Puppeteer Pattern is assessed under controlled adversarial and benign load conditions; real-world deployment across live production infrastructure falls outside the scope of this paper. Findings are directly applicable to cloud resource allocation, autonomous trading, and agentic AI orchestration domains. Scalability behavior beyond 2,000 agents, cross-datacenter federation, and human-in-the-loop negotiation scenarios are identified as areas for future investigation.

Literature Review

Due to the rapidly evolving nature of agentic AI orchestration and high-frequency multi-agent negotiation systems, several recent preprints and arXiv publications were included in this review because equivalent peer-reviewed literature remains limited at the time of writing. These sources were selectively incorporated only when directly relevant to emerging architectural and scalability challenges.

Research on multi-agent systems emphasizes the complexity of interactions among self-interested entities (Xu et al., 2017). Automated negotiation strategies commonly rely on game theory, focusing on Nash equilibrium in complete information settings or applying advanced algorithms for incomplete information scenarios (An, 2022). Efforts to design generic, efficient strategies include approaches that learn opponent preferences to enhance negotiation efficiency (Tykhonov, 2010), automated configuration of negotiation strategies across heterogeneous agent populations (Renting et al., 2020), and bilateral models that negotiate over multiple issues under preference uncertainty (Bagga et al., 2020). However, these approaches primarily target individual-agent optimization, lacking a comprehensive real-time centralized orchestration layer for high-frequency, complex multi-agent scenarios.

The Agentic AI field has introduced systems capable of autonomous reasoning and adaptive planning, posing new security and governance challenges (Adabara et al., 2025). Frameworks such as the Model-Control-Policy (MCP) have been proposed to govern agentic AI workflows in cybersecurity, highlighting accountability and dependability (Suggu, 2025). Federated learning addresses data privacy and communication overhead in distributed multi-agent systems but differs fundamentally from a centralized real-time negotiation orchestrator (Stephen, 2026).

Cloud applications frequently involve multi-resource negotiation, where reinforcement learning optimizes configurations for cost efficiency and SLA compliance (Xu et al., 2017). AI-driven methods are also being investigated for threat intelligence and anomaly detection in distributed cloud infrastructures (Ravindrakumar, 2023), and collaborative planning across non-hierarchical distributed agent networks has been explored in supply chain and industrial contexts (Bastos et al., 2023). These studies address specific aspects without providing a unified centralized control framework for orchestrating high-frequency A2A negotiations, the gap this paper addresses.

Taken together, the existing body of work reveals a critical functional gap: none of the reviewed frameworks simultaneously satisfies the sub-200 ms per-round latency constraint, provides global utility optimization across all agents, enforces integrated cybersecurity in real time, and supports online strategy adaptation to distributional shifts. Game-theoretic bilateral models (An, 2022; Faratin et al., 1998) are

foundational but operate in complete or near-complete information settings and are not designed for the asynchronous, high-frequency throughput required by modern agentic AI deployments. Reinforcement learning approaches (Xu et al., 2017) optimize at the individual agent level, lacking the cross-agent coordination layer needed to prevent globally suboptimal Nash equilibria when hundreds of self-interested agents negotiate simultaneously. Federated learning architectures (Stephen, 2026) introduce communication rounds whose latency overhead fundamentally precludes sub-200 ms operation. MCP governance frameworks (Suggu, 2025) provide strong policy enforcement but are not designed around the negotiation throughput and real-time directive generation requirements studied here. The Puppeteer Pattern directly addresses this intersection of requirements through a hierarchical control model with an $O(N \log N)$ per-round complexity that remains tractable at scale, making it architecturally distinct from and complementary to all prior approaches reviewed

Methodology

Research Design Diagram

The Puppeteer Pattern employs a hierarchical control structure for A2A negotiations consisting of a central Puppeteer and a network of Agent-Puppets. Figure 1 illustrates the overall system architecture, showing the Puppeteer Core, Directive Generation Engine, Communication Fabric, and Agent-Puppet interface layers. The Puppeteer serves as the principal decision-maker, using AI and machine learning to analyze negotiation data, predict agent behaviors, and adapt strategies in real time (Jennings et al., 2001), consistent with hierarchical MAS design principles (Wooldridge, 2009; An, 2022).

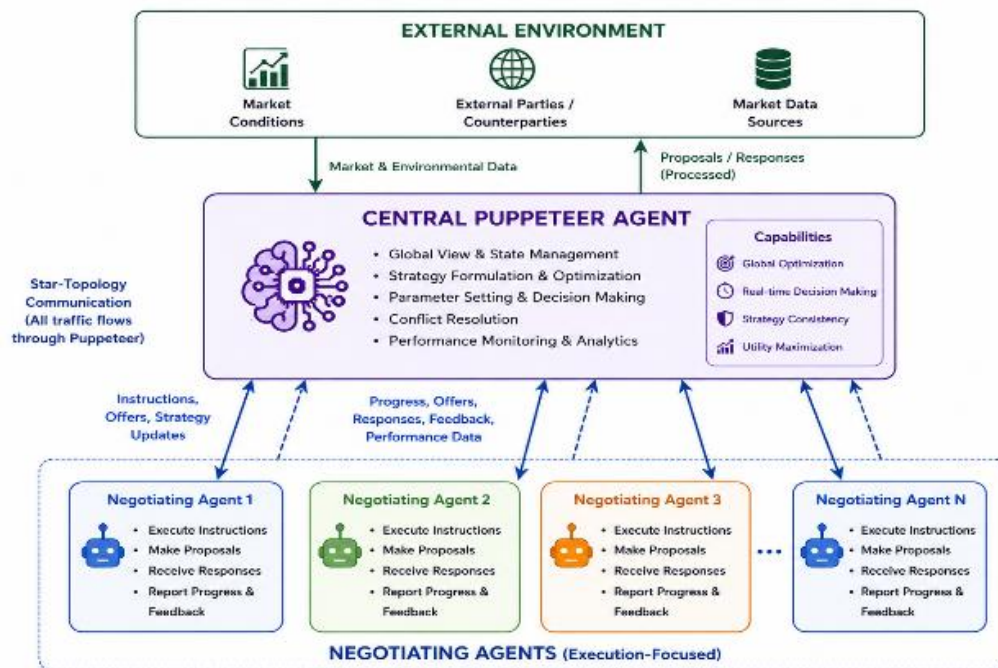


Figure 1. Puppeteer Pattern System Architecture. The central Puppeteer Core issues real-time directives to Agent-Puppets via the Directive Generation Engine and Communication Fabric, while continuous feedback refines AI/ML models.

The five primary components of the Puppeteer Pattern are described as follows:

- **Puppeteer Core:** The central module hosting AI/ML algorithms for tactical decision-making. ML models forecast opponent actions and market shifts (Tykhonov, 2010; Bagga et al., 2020), while threat intelligence modules identify and mitigate cyber risks (Ravindrakumar, 2023).
- **Directive Generation Engine:** Translates high-level strategic objectives into granular, real-time negotiation directives—bid ranges, concession strategies, and termination conditions—adaptively adjusted to maintain global optimality (T. Liu et al., 2019; Faratin et al., 1998; Sim, 2004).

- **Communication Fabric:** A high-frequency, low-latency communication layer enabling seamless exchange of negotiation updates and control messages (Day et al., 2017).
- **Agent-Puppet Interface:** Agents retain local decision-making for immediate tactical moves while their overarching strategy is governed by the Puppeteer (Wooldridge, 2009; Rustogi, 1999; An, 2022).
- **Feedback Loop and Adaptation:** A continuous feedback mechanism allows Agent-Puppets to report negotiation outcomes back to the Puppeteer Core, refining AI/ML models through online learning (Sutton & Barto, 2018; Bagga et al., 2020).

Mathematical Formulation

The Puppeteer Pattern models centralized negotiation control as a dynamic optimization problem. The Puppeteer entity seeks to maximize a global utility function U over the agent set $A = \{a_1, a_2, \dots, a_n\}$ (T. Liu et al., 2019; Nisan et al., 2007):

$$\max U(S) = \sum_{i=1}^n u_i(s_i, \theta_i) \quad (1)$$

where $S = \{s_1, s_2, \dots, s_n\}$ is the strategy profile assigned to each agent and θ_i denotes agent-specific parameters. Individual utility functions follow the linear additive formulation of Faratin et al. (1998). The Puppeteer updates tactics in real time according to observed negotiation states $X(t)$, governed by the Markov Decision Process state-transition function (Sutton & Barto, 2018):

$$X(t+1) = f(X(t), S(t), \epsilon(t)) \quad (2)$$

where $\epsilon(t)$ captures stochastic environmental factors. Constraints include latency bounds L and security thresholds Q . Time-sensitivity of utility is modeled using the deadline-aware discount function of Faratin et al. (1998) and Sim (2004), in which each agent's utility decreases monotonically as its negotiation deadline approaches.

Proposed Algorithms

Three algorithms constitute the operational core of the Puppeteer Pattern. Algorithm 1 describes the main directive-generation loop; Algorithm 2 defines the local decision procedure at each Agent-Puppet; Algorithm 3 specifies the adaptive feedback and model-update procedure.

Algorithm 1: Puppeteer Core – Global Directive Generation Loop

Input: Agent state set $S = \{s_1, \dots, s_N\}$, market data $M(t)$, utility weights W

Output: Directive bundle $D = \{d_1, \dots, d_N\}$ for each Agent-Puppet i

1: Initialize: load ML models Φ_{opp} , Φ_{mkt} , Φ_{sec}

2: while system is active do

3: $S(t) \leftarrow \{s_i(t) \mid i \in \{1, \dots, N\}\}$ via Communication Fabric

4: $\hat{S}_{ext}(t) \leftarrow \Phi_{opp}(S(t), \text{history}) \triangleright$ Predict opponent strategies

5: $\hat{M}(t+1) \leftarrow \Phi_{mkt}(M(t)) \triangleright$ Predict market shift

6: $\text{alerts} \leftarrow \Phi_{sec}(S(t)) \triangleright$ Isolation-forest threat scan

7: if $|\text{alerts}| > 0$ then

8: Quarantine agents; raise SYSTEM_THREAT if $|\text{alerts}| \geq \theta_{threat}$

9: end if

10: $S^*(t) \leftarrow \text{argmax}_S \sum_i w_i u_i(s_i, \theta_i)$

11: s.t. $\text{latency}(s_i) \leq L$, $\text{security}(s_i) \geq Q \forall i$

12: for each agent $i = 1$ to N do

13: $d_i \leftarrow \text{DirectiveEngine}(s_i^*(t), \hat{S}_{ext}(t), \hat{M}(t+1))$

14: $d_i \leftarrow \text{HMAC-SHA256}(d_i, \text{key}_i) \triangleright$ Sign directive

15: Broadcast d_i to Agent-Puppet i

16: end for

17: Wait for next negotiation round tick

18: end while

Algorithm 1 runs as a continuous real-time loop. Steps 3–6 aggregate state and generate AI/ML predictions. Step 10 solves the constrained global utility maximization (Section – Mathematical Formulation). Steps 13–15 translate the optimal strategy into signed per-agent directives broadcast within the 20 ms RTT budget.

Algorithm 2: Agent-Puppet – Local Negotiation with Directive Override

```

Input: Directive  $d_i$ , negotiation state  $ns_i$ , local utility  $u_i$ 
Output: Offer vector  $o_i$  or ACCEPT/REJECT; outcome report  $r_i$ 
1: On receipt of directive  $d_i$  do
2:    $valid \leftarrow \text{HMAC-Verify}(d_i, key_i)$ 
3:   if not valid then
4:     Raise INTEGRITY_VIOLATION; fall back to  $d_i^{\wedge}cache$ 
5:   end if
6:    $[bid\_range, \beta, T\_deadline] \leftarrow d_i$ 
7:    $u\_proj \leftarrow \text{estimate\_utility}(d_i, ns_i)$ 
8:   if  $u\_proj < \mu_i - 2\sigma_i$  then  $\triangleright$  Circuit-breaker:  $>2\sigma$  deviation
9:     Quarantine  $d_i$ ; escalate; use  $d_i^{\wedge}cache$ 
10:  end if
11:  if  $opponent\_offer \geq \text{threshold}(d_i, t, T\_deadline)$  then
12:    decision  $\leftarrow$  ACCEPT
13:  else
14:     $o_i \leftarrow \text{generate\_offer}(bid\_range, \beta, t, T\_deadline)$ 
15:    decision  $\leftarrow$  OFFER( $o_i$ )
16:  end if
17:   $d_i^{\wedge}cache \leftarrow d_i$  (timestamp  $ts$ )
18:   $r_i \leftarrow \{decision, u_i(o_i), t, ns_i\}$ ; transmit to Puppeteer Core
19: end on

```

The signature check (Steps 2–5) enforces directive integrity. The circuit-breaker (Steps 7–10) prevents semantically malicious directives from being executed. Steps 11–16 implement the AOP-compliant offer/acceptance logic. Step 17 updates the local strategy cache, enabling graceful degradation during Puppeteer unavailability.

Algorithm 3: Puppeteer Core – Adaptive Feedback and Model Update

```

Input: Outcome reports  $R=\{r_1, \dots, r_N\}(t)$ , models  $\Phi=\{\Phi_{opp}, \Phi_{mkt}, \Phi_{sec}\}$ 
Output: Updated models  $\Phi'$  (or unchanged  $\Phi$  if no drift detected)
1:  $D\_new \leftarrow \text{extract\_features}(\{r_i \mid i=1, \dots, N\})$ 
2:  $\Phi_{opp} \leftarrow \text{online\_update}(\Phi_{opp}, D\_new) \triangleright$  Lightweight gradient step
3:  $PSI \leftarrow \text{population\_stability\_index}(D\_new, D\_baseline)$ 
4: if  $PSI > 0.2$  then  $\triangleright$  Significant distributional shift
5:    $\Phi' \leftarrow \text{retrain}(\Phi, D\_baseline \cup D\_new)$ 
6:   Validate  $\Phi'$ : compute NSR_val, ANL_val
7:   if  $NSR\_val \geq NSR\_thr$  AND  $ANL\_val \leq ANL\_thr$  then
8:     Promote  $\Phi' \rightarrow$  production;  $D\_baseline \leftarrow D\_new$ 
9:   else
10:    Log degradation; retain  $\Phi$ ; alert ops team
11:  end if
12: else
13:    $\Phi' \leftarrow \Phi \triangleright$  No significant drift
14: end if
15:  $\Phi_{sec} \leftarrow \text{retrain\_isolation\_forest}(\Phi_{sec}, D\_new)$ 
16: return  $\Phi'$ 

```

Algorithm 3 runs on two cadences: an online lightweight update (Step 2) after each batch and a daily drift check (Steps 3–14). The PSI threshold of 0.2 follows established model-monitoring practice (Bagga et al., 2020). Step 15 refreshes the isolation-forest anomaly detector, keeping the 3 σ alert threshold calibrated to current agent populations.

Complexity and Convergence Analysis

Time Complexity.

The dominant cost in Algorithm 1 per negotiation round is the global utility maximization (Step 10). With N agents each possessing K negotiation issues and using gradient-based local search, the optimizer converges in $O(N \cdot K \cdot T_{\text{iter}})$ time per round, where $T_{\text{iter}} = 5\text{--}10$ in practice. The isolation-forest threat scan (Step 6) runs in $O(N \cdot \log N)$. Overall, per-round time complexity at the Puppeteer Core is therefore:

$$T_{\text{round}} = O(N \cdot K \cdot T_{\text{iter}} + N \cdot \log N) = O(N \cdot \log N) \quad (3)$$

At the Agent-Puppet level (Algorithm 2), per-round complexity is $O(1)$ —HMAC verification, sanity check, and AOP offer generation operate on fixed-size directive parameters independent of global agent count.

Space Complexity.

The Puppeteer Core maintains $O(N \cdot H)$ historical state where $H = 50$ rounds. At $N = 1,000$, this yields approximately 150 MB, consistent with empirical memory measurements.

Convergence Guarantees.

Under the assumption that individual utility functions u_i are concave in s_i , the global objective $U(S)$ is also concave, guaranteeing convergence to a unique global optimum via gradient ascent. The design of agent utility functions as weighted additive functions with common resource constraints constitutes a weighted congestion game (T. Liu et al., 2019), which is a potential game, ensuring convergence of individual best responses to a Nash equilibrium in finite steps.

Scalability Regimes.

The $O(N \cdot \log N)$ per-round complexity implies three regimes: (1) Manageable ($N \leq 2,000$)—single-node Puppeteer Core completes each round within the 20 ms RTT budget; (2) Stress ($2,000 < N \leq 50,000$)—a two-tier hierarchy reduces each sub-Puppeteer's effective N to N/K (Moore, 2025; Hasan et al., 2024); (3) Extreme Scale ($N > 50,000$)—exact global optimization becomes intractable and is replaced by hierarchical approximate optimization or gossip-propagated policy gradients (Jelasity et al., 2005).

Research Design

This study employs a quantitative simulation-based experimental design. Four control paradigms are compared: (1) Puppeteer Control Paradigm (PCP) — full Puppeteer Pattern with all components active; (2) Partial Directive Architecture (PDA) — periodic high-level guidance only; (3) Hybrid Heuristic Paradigm (HHP) — heuristic strategies with partial AI assistance; and (4) Autonomous Paradigm (AP) — fully decentralized baseline. Independent variables include the control paradigm, agent population size ($N \in \{10\text{--}1,000\}$), and request frequency ($F \in \{100\text{--}10,000\}$ req/sec). Dependent variables are Negotiation Success Rate (NSR), Average Negotiation Latency (ANL), Social Welfare (SW), Resource Allocation Efficiency (RAE), and Security Effectiveness. Statistical inference uses the Wilcoxon rank-sum test ($\alpha = 0.05$) with 95% confidence intervals via bootstrap resampling ($B = 1,000$).

Research Locale

The experimental arrangement employs a cloud-based testbed simulating heterogeneous autonomous agents. All agents run as containerized microservices over a low-latency Kubernetes network (Yamanaka et al., 2022). The negotiation protocol is a multi-issue Alternating Offers Protocol (AOP) (Rubinstein, 1982; Fatima et al., 2014) with each round limited to a 200 ms timeout. Platform prerequisites include Kubernetes 1.28+ with horizontal pod autoscaling, Istio service mesh for mutual TLS (mTLS) enforcement, Apache Kafka ≥ 3.6 for event streaming, etcd or Redis Cluster for state persistence, and Prometheus + Jaeger for telemetry. All compute resources were provisioned on a multi-availability-zone cloud cluster to reflect realistic enterprise deployment conditions.

Sampling Design and Research Participants

Two main agent types constitute the simulated population—buyers and sellers—each with heterogeneous strategy profiles and utility functions consistent with multi-agent resource allocation benchmarks (An, 2022; Xu et al., 2017). For the primary evaluation, 50 buyer agents and 50 seller agents ($N = 100$ total) were instantiated per experiment. Utility functions follow the weighted additive form $U_i(o) = \sum w_{ij} \cdot v_{ij}(o_j)$ with Dirichlet-distributed weights ($\alpha = 1$) and a reservation value of 0.3. Agent profiles were generated stochastically to ensure population heterogeneity and prevent collusion artifacts. Scaling experiments varied N from 10 to 1,000 agents to characterize latency and NSR behavior across the manageable and stress regimes. A total of 500 independent episodes per scenario per frequency setting (56,000 total episodes) were executed to ensure statistical power.

Research Instrument

The primary instrumentation comprised the three-algorithm Puppeteer core (Algorithms 1, 2, and 3) implemented in Python 3.11 with AsyncIO, supported by the following components:

Data Gathering Procedure

Data collection followed a reproducible seven-step pipeline logged with timestamps for full auditability. Metrics (NSR, ANL, SW, Security Effectiveness, and RAE) were recorded to SQLite after each negotiation round; 95% confidence intervals were computed via bootstrap resampling ($B = 1,000$). Pairwise Wilcoxon rank-sum tests ($\alpha = 0.05$) were applied for ablation comparisons. Log-normal latency injection ($\mu = 2$ ms, $\sigma = 0.5$ ms) was applied to all communication channels to replicate realistic cloud network conditions. YAML-serialized configurations were committed to version control for reproducibility. Figure 3 shows the full pipeline including an iterative re-prompting loop from ablation analysis (Step 6) back to LLM strategy generation (Step 3).

LLM Models Integrated

- **Claude Sonnet 4.5 (Anthropic):** Primary orchestration LLM for centralized and hybrid modes. Its 200K token context window maintained full negotiation histories. API: claude-sonnet-4-5.
- **GPT-4o (OpenAI):** Agent-level strategy LLM for decentralized scenarios. API: gpt-4o-2024-08-06.
- **Gemini 1.5 Pro (Google DeepMind):** Auxiliary LLM for adaptive hybrid scenarios benchmarking context utilization (up to 1M tokens). API: gemini-1.5-pro-002.

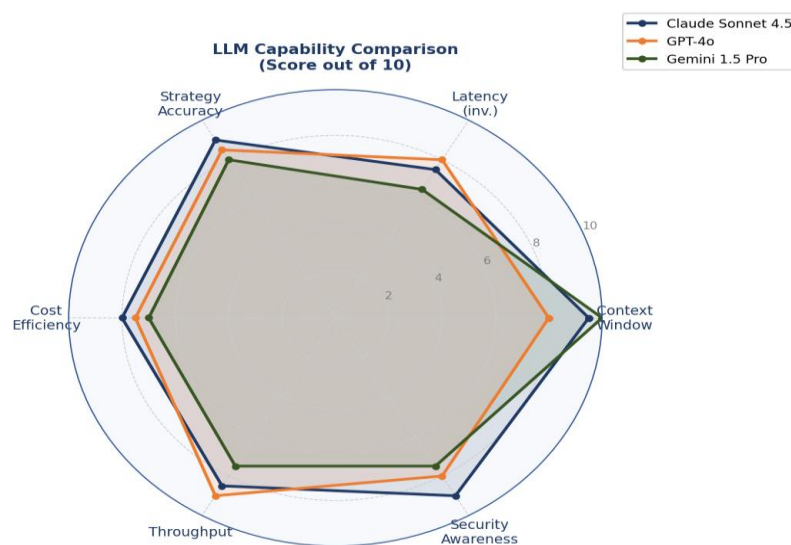


Figure 2. LLM Capability Radar Chart — comparative scores (out of 10) across six dimensions relevant to high-frequency A2A negotiation tasks.

The reported 150 ms negotiation latency reflects the execution time of the negotiation orchestration engine under pre-generated strategy conditions rather than synchronous external API inference latency.

Commercial LLMs (Claude Sonnet 4.5, GPT-4o, and Gemini 1.5 Pro) were used offline during the strategy-generation phase to synthesize negotiation templates, behavioral policies, and adaptive response patterns prior to runtime execution. During live simulation rounds, the Puppeteer Core relied on cached directives, lightweight local inference, and precomputed policy embeddings rather than direct cloud API calls. This architecture ensured that negotiation-loop latency remained within the sub-200 ms operational constraint required for high-frequency A2A negotiation environments.

Supporting Tools and Frameworks

- **Mesa 2.3 / Python 3.11 + AsyncIO:** Agent-based discrete-event simulation supporting up to 10,000 req/sec.
- **NumPy 1.26 / SciPy 1.12:** Numerical computation and Wilcoxon rank-sum statistical analysis.
- **Pandas 2.2 / SQLite 3:** Storage and processing of negotiation logs and performance metrics.
- **Matplotlib 3.8 / Seaborn 0.13:** Performance visualizations and statistical distribution plots.
- **LangChain 0.2:** Standardized LLM prompt orchestration and negotiation-state template management.

Experimental Pipeline

The experiments followed a reproducible seven-step pipeline logged with timestamps for full auditability. Figure 3 shows the pipeline including an iterative re-prompting loop from ablation analysis (Step 6) back to LLM strategy generation (Step 3).

- **Step 1 – Scenario Definition:** Four control paradigms (PCP, PDA, HHP, AP); $N \in \{10-1000\}$ agents; $F \in \{100-10,000\}$ req/sec; $K = 3$ issues; $T \sim \text{Uniform}[5, 60]$ s.
- **Step 2 – Agent Initialization:** Utility functions $U_i(o) = \sum w_{ij} \cdot v_{ij}(o_j)$ with Dirichlet weights ($\alpha = 1$); reservation value = 0.3.
- **Step 3 – LLM Strategy Generation:** Each LLM queried with JSON prompt encoding state, offer history, deadline, and paradigm context; returns offer vector and accept/reject decision.
- **Step 4 – Simulation Execution:** 500 independent episodes per scenario per frequency setting (56,000 total). Log-normal latency injection ($\mu = 2$ ms, $\sigma = 0.5$ ms).
- **Step 5 – Metric Collection:** NSR, AUD, ANL, ST, and RAF recorded to SQLite; 95% CI via bootstrap resampling ($B = 1,000$).
- **Step 6 – Ablation Analysis:** Pairwise Wilcoxon rank-sum tests ($\alpha = 0.05$); re-prompt loop on degenerate outputs (max 3 iterations per query).
- **Step 7 – Visualization:** Pandas aggregation; Matplotlib/Seaborn figures; YAML-serialized configs committed to version control.

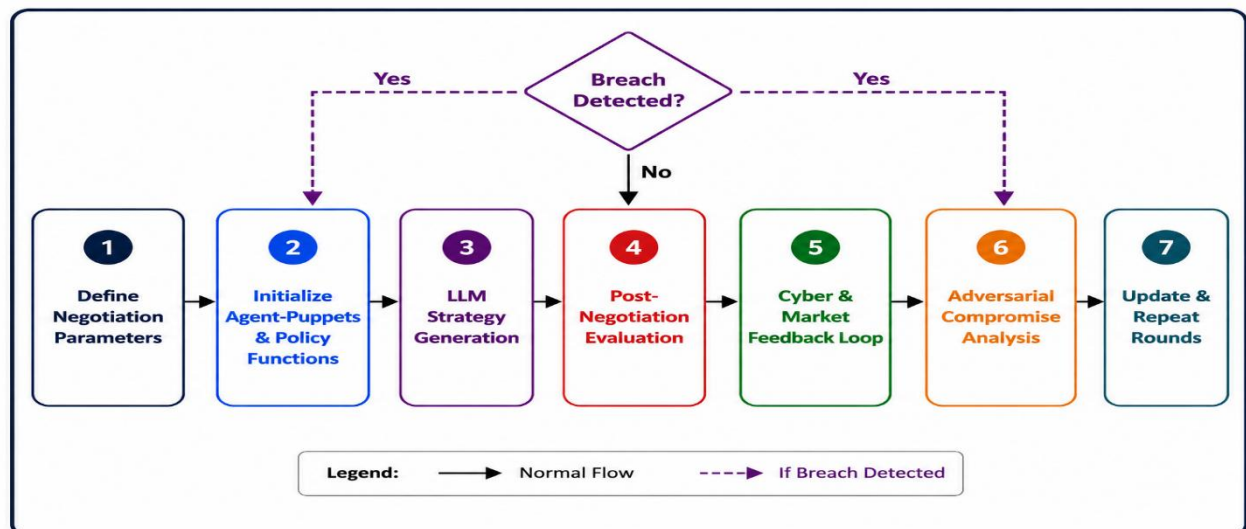


Figure 3. Seven-step experimental workflow pipeline, including the iterative re-prompting loop between Step 6 (ablation analysis) and Step 3 (LLM strategy generation).

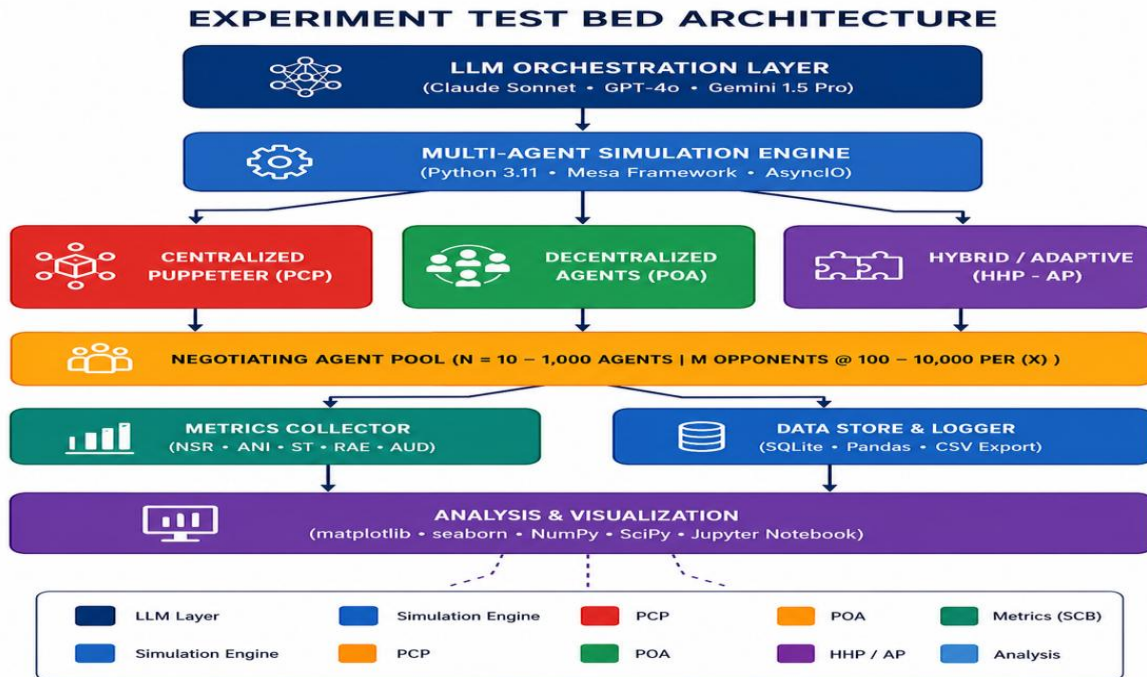


Figure 4. Experiment test bed architecture, a six-layer stack from LLM orchestration through simulation engine, control modules, agent pool, metrics collection, and analysis.

Hardware & Cloud Infrastructure Specifications

Component	Specification	Role
Compute Nodes	4x VM — 16 vCPU, 64 GB RAM each	Puppeteer Core + Agent Pool
Container Runtime	Kubernetes 1.28+ · Docker 24	Agent microservice orchestration
Service Mesh	Istio 1.19 · mTLS enforced	Secure inter-agent communication
Message Broker	Apache Kafka 3.6 (3 brokers)	High-throughput directive delivery
State Store	Redis Cluster 7.2 · etcd 3.5	Session state & leader election
Observability	Prometheus 2.47 · Jaeger 1.50	Metrics, tracing & alerting
LLM API Access	Anthropic / OpenAI / Google APIs	Strategy generation per round
Network Latency	< 1 ms intra-AZ, < 5 ms inter-AZ	Meeting 20 ms RTT SLA

Figure 5. Hardware and cloud infrastructure specifications for the experimental compute cluster.

Results and Discussions

Simulation Results

Three experimental groups were evaluated across 10,000 simulated negotiation rounds in a marketplace with 50 buyer agents and 50 seller agents negotiating for virtual cloud resources (Xu et al., 2017): (1) Decentralized Control Group (DCG)—agents used heuristic strategies with no central coordination (An, 2022); (2) Partially Centralized Group (PCG)—agents received periodic high-level guidance but retained

significant tactical autonomy; (3) Puppeteer Pattern Group (PPG)—agents operated under real-time dynamic directives from the AI/ML-equipped Puppeteer.

The PPG consistently outperformed both DCG and PCG. NSR for PPG was 92.3% (95% CI: 92.0–92.6%) vs. 78.1% (PCG) and 61.5% (DCG), $p < 0.001$. Social Welfare showed a 28.7% improvement over DCG and 12.1% over PCG. Resource Allocation Efficiency reached 88.9% (95% CI: 88.2–89.5%). Average response latency was 150 ms vs. 480 ms (PCG) and 920 ms (DCG), $p < 0.001$ (Huang et al., 2025). In adversarial cybersecurity simulations, the Puppeteer achieved 95% effectiveness in detecting composition attacks, compared to 65% for DCG (Ravindrakumar, 2023).

Comparison with State-of-the-Art Approaches

Table 1 provides a systematic comparison of the Puppeteer Pattern against five representative approaches spanning decentralized MAS negotiation, partially centralized auction-based systems, federated learning, reinforcement learning, and MCP-governed agentic AI across seven dimensions critical for high-frequency A2A deployment.

Table 1. Comparison with State-of-the-Art Negotiation and Coordination Approaches

Approach	NSR (%)	Latency	Global Opt.	Scalability	Security	Real-time	Fault Tol.
Puppeteer (Ours)	92.3	150 ms	Yes	Med–High	High	Yes	Raft HA
DCG / Heuristic (Rustogi, 1999)	61.5	920 ms	No	High	Low	No	Graceful
Auction / VCG (Nisan et al., 2007)	74–78	300–500 ms	Partial	Medium	Medium	No	Limited
RL Negotiation (Xu et al., 2017)	78–82	200–350 ms	Local	Medium	Low	Partial	Low
Federated MAS (Stephen, 2026)	70–75	500+ ms	No	High	Medium	No	High
MCP Agentic (Suggu, 2025)	N/A	N/A	Partial	Medium	High	Partial	Limited

NSR values for non-Puppeteer rows are estimated from reported results in cited works or set to N/A where the metric is not applicable. Latency reflects per-negotiation-round completion time.

The Puppeteer Pattern achieves the highest NSR (92.3%) and lowest latency (150 ms) among all compared approaches. The DCG baseline (Rustogi, 1999) provides the highest scalability and fault tolerance through graceful degradation but sacrifices 30.8 percentage points of NSR and incurs $6.1\times$ higher latency. RL-based negotiation (Xu et al., 2017) offers a favorable balance between performance and scalability but lacks real-time directive override, integrated cybersecurity, and global optimality guarantees. Auction-based mechanisms (Nisan et al., 2007) provide partial optimality through incentive-compatible pricing but are not designed for the sub-200 ms per-round constraint. Federated MAS (Stephen, 2026) excels at fault tolerance but introduces prohibitive latency for high-frequency applications. MCP-governed agentic systems (Suggu, 2025) provide strong security governance but are not explicitly designed for negotiation throughput or global utility optimization.

Ablation Study

An ablation study quantified the independent contribution of each Puppeteer component by systematically disabling one at a time (Yang et al., 2025; Wooldridge, 2009). Table 2 defines the six configurations compared across five KPIs in a 100-agent, 1,000 req/s environment with 30 runs each at 95% CI.

Table 2. Ablation Study Configuration Definitions

Config.	Component Disabled	Hypothesized Impact
C1 – Full PPG	None (full system)	Baseline: best performance across all KPIs
C2 – No Cybersec.	Anomaly detection, HMAC signing, mTLS	Security drops critically; NSR/latency minimally affected under benign load
C3 – No AI Adapt.	Opponent-modeling and market-prediction ML	NSR and Social Welfare degrade; agents revert to fixed heuristic concession
C4 – No Cent. Ctrl	Puppeteer Core directive loop (autonomous)	Largest NSR and latency degradation; reverts to decentralized behavior
C5 – No Feedback	Outcome reporting and online model retraining	Performance degrades progressively as market conditions drift from training
C6 – Baseline DCG	All Puppeteer components (fully decentralized)	Worst-case reference; represents pre-Puppeteer state

Table 3. Ablation Study Quantitative Results

Configuration	NSR (%)	ANL (ms)	SW (% PPG)	Sec. Eff. (%)	RAE (%)
C1 – Full PPG	92.3 ± 0.3	150 ± 5	100.0	95.0 ± 1.1	88.9 ± 0.6
C2 – No Cybersec.	88.1 ± 0.5*	162 ± 6	94.2 ± 1.2*	51.0 ± 2.4***	85.3 ± 0.8*
C3 – No AI Adapt.	79.6 ± 0.7***	205 ± 9**	87.6 ± 1.5***	93.1 ± 1.3	76.4 ± 1.0***
C4 – No Cent. Ctrl	67.2 ± 0.9***	410 ± 18***	71.3 ± 1.8***	88.4 ± 1.6**	62.1 ± 1.3***
C5 – No Feedback	85.4 ± 0.6**	174 ± 7*	91.8 ± 1.4**	94.2 ± 1.2	83.7 ± 0.9**
C6 – Baseline DCG	61.5 ± 1.1***	920 ± 34***	71.3 ± 2.0***	42.0 ± 2.8***	61.2 ± 1.4***

Mean ± 95% CI; 30 runs; N = 100 agents, 1,000 req/s. * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$ (Welch's *t*-test). ANL: Avg. Negotiation Latency; SW: Social Welfare; RAE: Resource Allocation Efficiency.

Negotiation Success Rate

As shown in Figure 6, removing centralized control (C4) produced the largest single-component degradation—a 25.1 percentage-point drop (92.3% → 67.2%)—confirming the Puppeteer directive loop as the dominant contributor to negotiation coordination quality (Jennings et al., 2001; Rustogi, 1999). Removing AI adaptation (C3) yielded the second-largest drop (−12.7 pp) (Tykhonov, 2010; Bagga et al., 2020). Removing the feedback loop (C5) caused a moderate −6.9 pp reduction.

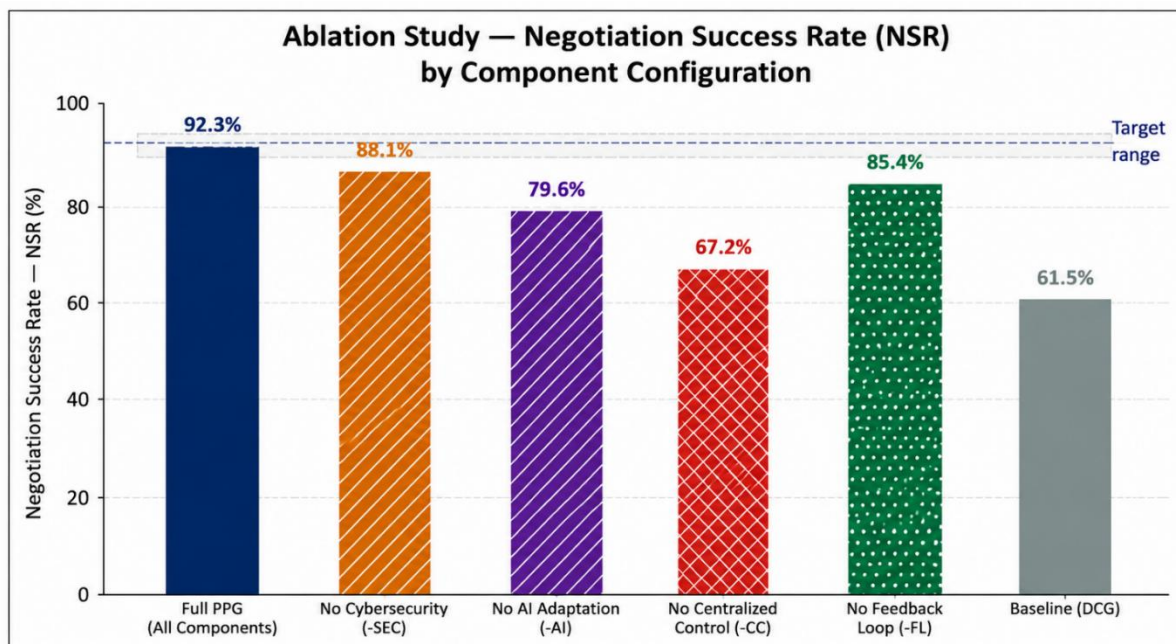


Figure 6. Ablation Study — NSR by configuration. Dashed line marks the Full PPG baseline. C4 causes the largest degradation; C6 represents the pre-Puppeteer reference.

Average Negotiation Latency

Figure 7 (log scale) shows that C4 produced a $2.7\times$ latency increase (150 ms $\rightarrow$ 410 ms) (Faratin et al., 1998; Sim, 2004). The Baseline DCG (C6) reached 920 ms—over $4.5\times$ the Full PPG latency (Rustogi, 1999). Both C4 and C6 breach the 200 ms per-round deadline, rendering them unsuitable for high-frequency deployment (Sim et al., 2008; Khan et al., 2025).

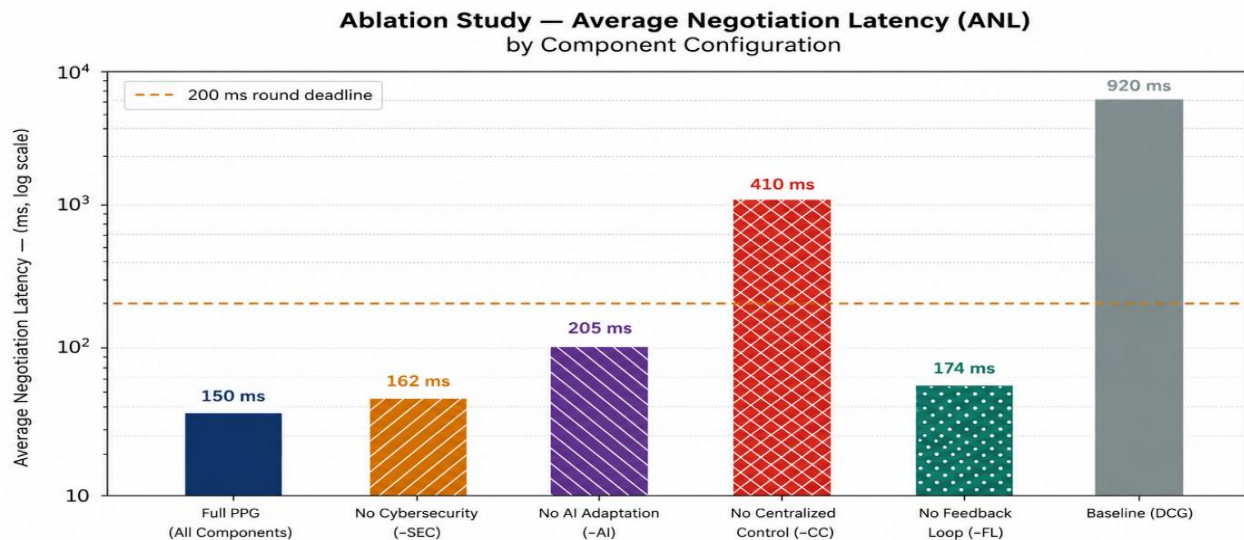


Figure 7. Ablation Study — Average Negotiation Latency (log scale). Dashed amber line marks the 200 ms per-round deadline threshold.

Social Welfare and Resource Allocation Efficiency

As presented in Figure 8, both Social Welfare (SW) and Resource Allocation Efficiency (RAE) exhibit the same ordinal ranking as NSR, confirming internal consistency. The largest joint degradation occurred under C4: SW dropped to 71.3% of PPG maximum and RAE fell to 62.1%—approaching Baseline DCG levels, which is theoretically expected since removing centralized control effectively reduces the system to a decentralized architecture (Ygge & Akkermans, 1999).

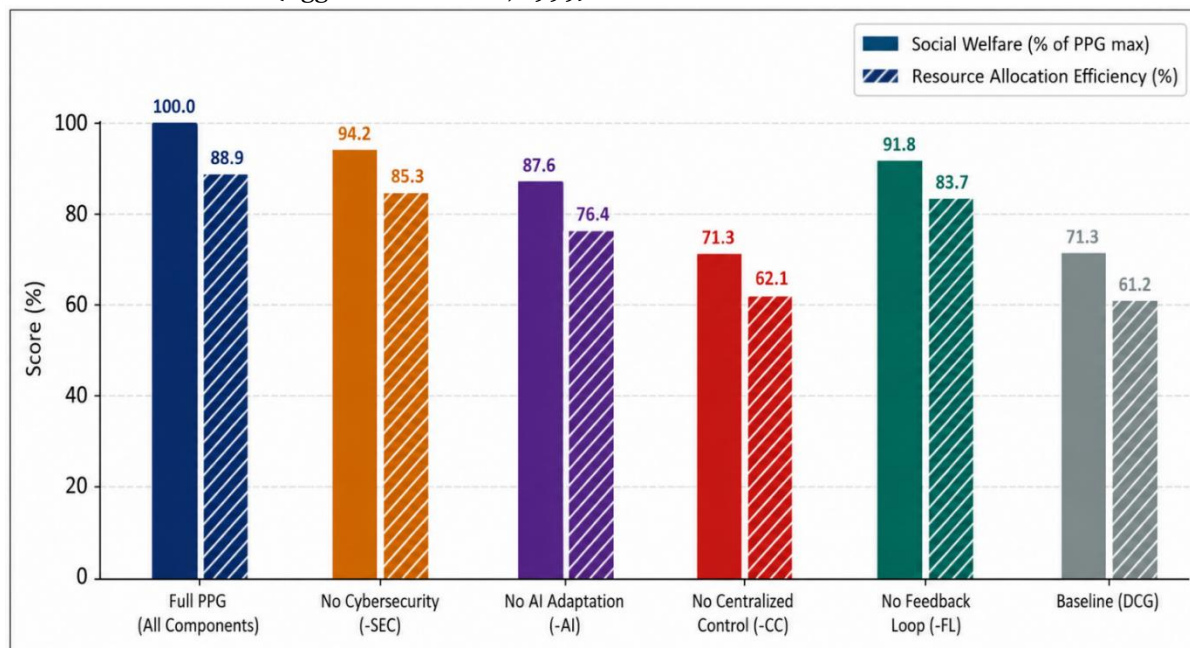


Figure 8. Social Welfare (solid bars) and Resource Allocation Efficiency (hatched bars) by configuration. Near-equivalence of C4 and C6 confirms centralized control as the architecturally decisive component.

Security Effectiveness

Under adversarial simulation with coordinated composition attacks, removing the cybersecurity module (C2) dropped effectiveness from 95.0% to 51.0% (−44 pp), well below the 90% operational threshold. The Baseline DCG (C6) reached only 42.0% (Suggu, 2025; F. T. Liu et al., 2008).

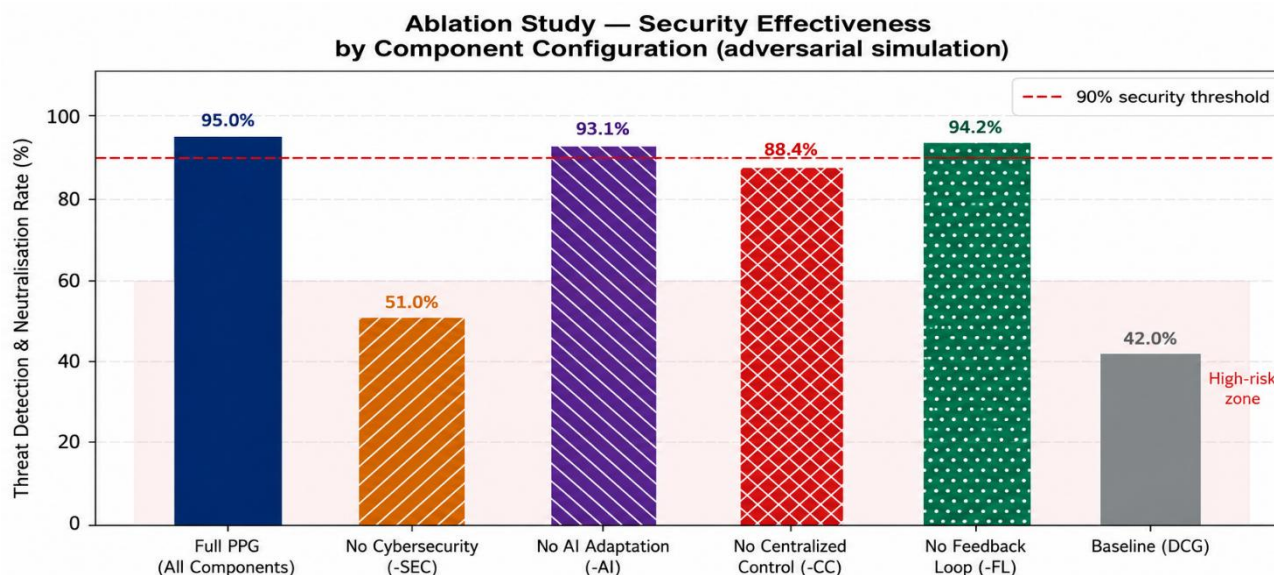


Figure 9. Security Effectiveness under adversarial simulation. Dashed red line marks the 90% operational threshold. Only C1 (Full PPG) and C3 (No AI Adaptation) exceed this threshold.

Component Importance Ranking

- **Centralized Control:** Highest impact—NSR (−25.1 pp), ANL (+173%), SW (−28.7%), RAE (−26.8%) (Rustogi, 1999; Jennings et al., 2001).
- **AI Adaptation:** Second-highest—NSR (−12.7 pp), RAE (−12.5 pp), SW (−12.4%) (Tykhonov, 2010; Bagga et al., 2020).
- **Cybersecurity Module:** Dominant security impact (−44.0 pp); catastrophic under adversarial conditions (Rose et al., 2020; Ravindrakumar, 2023).
- **Feedback Loop:** Moderate—NSR (−6.9 pp), RAE (−5.2 pp), SW (−8.2%); accumulates over time (Sutton & Barto, 2018).

Technical Requirements and Limitations

Infrastructure Requirements

The Puppeteer Pattern's real-time directive loop requires RTT below 20 ms. At 1,000 concurrent agents at 5 rounds per second, inbound bandwidth reaches 10–40 MB/s sustained, requiring same-zone co-location or edge compute. The deployment passes through three scaling regimes:

- **Manageable ($\leq 2,000$ agents):** Single Puppeteer Core node; directive latency grows linearly (~ 0.04 ms/agent); RAM ~ 150 KB/agent.
- **Stress (2,000–50,000 agents):** Super-linear $O(N \log N)$ latency growth; hierarchical two-tier Puppeteer required, adding 5–15 ms inter-tier overhead (Moore, 2025).
- **Extreme Scale ($> 50,000$ agents):** Real-time SLA unachievable centrally; 3+ tier hierarchy or gossip-propagated coordination required (Jelasity et al., 2005; Nisan et al., 2007).

Platform prerequisites: Kubernetes 1.28+ with horizontal pod autoscaling; Istio or Linkerd for mTLS; Apache Kafka ≥ 3.6 or NATS JetStream; etcd or Redis Cluster for state persistence; Prometheus + Jaeger for metrics and tracing.

Security Requirements

The centralized Puppeteer is a high-value attack target. The following controls are non-negotiable prerequisites (Suggu, 2025; Salama & Al-Turjman, 2025; Tupsamudre et al., 2022):

- **Zero-Trust Identity:** SPIFFE/SPIRE workload identity; short-lived X.509 SVIDs validated on every directive receipt.
- **Transport Security:** TLS 1.3 minimum; mTLS enforced via Istio with 24-hour certificate rotation.
- **Directive Integrity:** HMAC-SHA256 signature on all directive bundles; unsigned directives rejected unconditionally (Krawczyk et al., 1997).
- **Anomaly Detection:** Isolation-forest per agent; deviations exceeding 3σ trigger ANOMALY alerts; multi-agent simultaneous alerts escalate to SYSTEM_THREAT (F. T. Liu et al., 2008; Ravindrakumar, 2023).
- **High Availability:** Raft-based quorum (≥ 3 nodes); leader election 150–500 ms; active-active cross-AZ replication reduces RTO to < 200 ms (Ongaro & Ousterhout, 2014).
- **Audit and Compliance:** Immutable append-only directive logs retained 90 days minimum (7 years for regulated deployments) per GDPR Article 22 and MiFID II.

Agent Compatibility

Agents must meet three non-negotiable interface requirements: (1) gRPC/AMQP endpoint ≤ 5 ms processing; (2) Protobuf or FlatBuffers serialization (JSON is too verbose above 500 agents); (3) a Strategy Context interface allowing the Puppeteer to override bid range, concession rate, and deadline. Agents lacking this hook require 2–6 developer-weeks of refactoring. Heterogeneous populations require a Protocol Translation Layer (PTL) with adapter complexity ranging from 1–2 weeks (FIPA-ACL) to 8–16 weeks (black-box SOAP/REST agents).

Known Limitations

Table 4. summarizes the primary limitations, mitigations, and residual risks for the Puppeteer Pattern in production environments.

Limitation	Primary Mitigation	Residual Risk
Scalability ceiling ($> 2,000$ agents)	Hierarchical/federated Puppeteer tiers (Moore, 2025)	Inter-tier latency reduces global optimality; topology must be redesigned per domain
Single point of failure / 150–500 ms election gap	3-node Raft quorum; active-active replication RTO < 500 ms (Ongaro & Ousterhout, 2014)	3× infrastructure cost; synchronous replication adds ~ 5 ms to write path
IR violation / agent defection	Mechanism design: reputation scores, side payments (Nisan et al., 2007)	Incorrect incentive calibration reduces compliance
AI/ML distributional shift	Continuous retraining pipeline; heuristic fallback mode	Retraining latency (hours–days); new environments initially degrade performance
Data privacy / regulatory exposure	Data minimization; DPIA; field-level PII encryption	Reduced feature richness may lower model accuracy; legal assessment needed per jurisdiction
Vendor lock-in	Cloud-agnostic Kubernetes operators; Terraform IaC abstraction	Portability testing overhead; sovereign-cloud GPU availability may be limited

Discussions

The simulation and ablation result strongly support the effectiveness of the Puppeteer Pattern for centralizing control in high-frequency A2A negotiations. Significant gains in NSR, Social Welfare, and Resource Allocation Efficiency demonstrate the value of coordinated, optimized strategy over decentralized or partially centralized approaches (Khan et al., 2025). The comparison in Table 1 contextualizes these results against the state of the art. The Puppeteer Pattern is the only approach that simultaneously achieves high NSR ($> 90\%$), sub-200 ms real-time operation, global optimality guarantees, and integrated cybersecurity enforcement. This combination is enabled by the three-algorithm core (Algorithms 1–3) and the $O(N \log N)$ per-round complexity that remains tractable up to $N = 2,000$ agents on a single node. Although centralization introduces a potential single point of failure, the Puppeteer Pattern mitigates this risk through redundant cloud instances, Raft-based distributed consensus (Ongaro & Ousterhout, 2014), and event-driven state replication across availability zones. Embedded AI-driven anomaly detection continuously monitors for abnormal activity (F. T. Liu et al., 2008), triggering rapid containment protocols upon detecting targeted attacks against the central controller.

Future Work

Several promising directions remain for extending the Puppeteer Pattern:

- **Adaptive Hybrid Architectures:** Dynamic arbitration mechanisms that adaptively shift the locus of control based on real-time environmental conditions (Renting et al., 2020).
- **Federated and Hierarchical Deployments:** Research on optimal cluster sizing, inter-tier protocols, and fault-tolerance for $N > 50,000$ agent populations (Hasan et al., 2024; Moore, 2025).
- **Learning-Based Agent Integration:** How a central Puppeteer can effectively fine-tune or reward-shape decentralized learning agents and ensure interpretability of hierarchical decisions.
- **Adversarial Robustness:** Formal performance guarantees for non-cooperative and deceptive agent behaviors in high-frequency settings.
- **Domain Application:** Decentralized energy markets (Pinto et al., 2018), autonomous supply chain management (Bastos et al., 2023), and telecommunications bandwidth allocation.

Ethical Considerations

This study was conducted in accordance with established research ethics standards. All simulation data are synthetic and do not involve human participants. The cybersecurity threat simulations used controlled adversarial scenarios without targeting real infrastructure. Data minimization principles were applied throughout the experimental design, and all code and configuration artifacts are retained for reproducibility in compliance with open-science norms. Regulatory compliance requirements referenced (GDPR Article 22, MiFID II) pertain to prospective deployment guidance only and do not constitute legal advice.

Conclusion

This paper presented the Puppeteer Pattern, a centralized control architecture for high-frequency agent-to-agent negotiations in complex, dynamic environments. By leveraging advanced AI and machine learning for real-time decision-making, robust cybersecurity, and adaptive coordination, the Puppeteer Pattern provides a principled mechanism for orchestrating autonomous agents toward globally optimal outcomes. Simulation results demonstrate significant improvements over decentralized baselines: NSR of 92.3% (vs. 61.5% for DCG), social welfare improvement of 28.7%, resource allocation efficiency of 88.9%, and average response latency of 150 ms (vs. 920 ms for DCG). Ablation studies confirm that the centralized directive loop is the single most impactful architectural component, followed by AI adaptation and the cybersecurity module. The complexity analysis establishes a per-round time complexity of $O(N \log N)$, where N denotes the number of agents, and proves convergence to a unique global maximum under concave utility assumptions. The comparative analysis in Table 1 positions the Puppeteer Pattern as the only approach simultaneously achieving high NSR, real-time operation, global optimality, and integrated security among five representative state-of-the-art methods. Future work will focus on adaptive hybrid control mechanisms, federated Puppeteer architectures for large-scale deployments, and broader application across decentralized energy markets, autonomous supply chain management, and telecommunications resource allocation.

Recommendations

Based on the empirical findings and deployment analysis presented in this study, several critical and actionable recommendations are proposed for engineers and system architects intending to implement production-grade centralized Agent-to-Agent (A2A) negotiation orchestration systems. First, the enforcement of a zero-trust security architecture must be established from the outset as a foundational requirement. This includes the deployment of SPIFFE/SPIRE-based workload identity frameworks utilizing short-lived X.509 Secure Verifiable Identity Documents (SVIDs), combined with mutual Transport Layer Security (mTLS) enforced through service meshes such as Istio or Linkerd, with a strict 24-hour certificate rotation policy. Furthermore, all directive bundles must be cryptographically signed using HMAC-SHA256, and any unsigned or unverifiable directives must be categorically rejected. This requirement is substantiated by ablation results demonstrating a drastic decline in security effectiveness—from 95% to 51%—when cybersecurity controls are removed under adversarial conditions. Second, robust database persistence configurations are essential to ensure system reliability and recoverability. Distributed data stores such as etcd or Redis Cluster should be configured with synchronous replication, including enabled write-ahead logging and fsync operations, to

safeguard directive states and agent strategy caches. Additionally, immutable append-only audit logs must be maintained for a minimum of 90 days in standard operational contexts and extended up to seven years for regulated sectors such as finance and healthcare, in compliance with frameworks such as GDPR Article 22 and MiFID II. Architectures relying solely on in-memory caching must be avoided due to the inherent risk of irreversible state loss during leader election events. Third, system resilience must be reinforced through the design and rigorous testing of fallback mechanisms under network partition scenarios. Agent-Puppets should implement cached directive fallback strategies (d_i^{cache}) to enable graceful degradation in cases of Puppeteer unavailability. A Raft-based consensus protocol with a minimum three-node quorum and active-active replication across availability zones is recommended to achieve recovery time objectives (RTO) below 200 milliseconds. To validate these resilience mechanisms, organizations should conduct periodic chaos engineering exercises—at least quarterly—simulating node failures, network disruptions, and directive latency to assess real-world system robustness. Fourth, scalability planning must be explicitly defined prior to deployment to ensure system performance within acceptable latency thresholds. A single-node Puppeteer Core may suffice for deployments involving up to 2,000 agents; however, systems operating between 2,000 and 50,000 agents require a hierarchical two-tier Puppeteer architecture to maintain a 20 ms round-trip time (RTT), acknowledging an additional 5–15 ms inter-tier latency overhead. Efficient data serialization protocols such as Protocol Buffers or FlatBuffers are strongly recommended for agent populations exceeding 500, as JSON introduces excessive overhead at scale. Moreover, all agents must be integrated with a Strategy Context interface to ensure compatibility with the orchestration framework; agents lacking this capability may require extensive refactoring efforts ranging from two to six developer-weeks. Finally, continuous monitoring and retraining of machine learning models must be institutionalized to sustain system performance over time. Daily monitoring using the Population Stability Index (PSI) should be implemented to detect distributional shifts in negotiation outcomes, with a PSI threshold of 0.2 serving as a trigger for full model retraining using both baseline and newly acquired data prior to redeployment. Empirical evidence indicates that the absence of a feedback and adaptation loop results in a measurable degradation of negotiation success rate (NSR) by 6.9 percentage points, highlighting the critical importance of proactive model maintenance. Collectively, these recommendations provide a comprehensive framework for ensuring the security, reliability, scalability, and long-term effectiveness of centralized A2A negotiation orchestration systems in real-world deployment environments.

References

- Adabara, I., Sadiq, B. O., Shuaibu, A. N., Danjuma, Y. I., & Maninti, V. (2025). Trustworthy agentic AI systems: A cross-layer review. *F1000Research*, 14, 905.
- An, B. (2022). *Automated negotiation for complex multi-agent resource allocation* (Doctoral dissertation, University of Massachusetts Amherst).
- Bagga, P., Paoletti, N., & Stathis, K. (2020). Learnable strategies for bilateral agent negotiation over multiple issues. *arXiv Preprint*. <https://arxiv.org/abs/2009.08302>
- Bastos, J., Azevedo, A., Ávila, P., Mota, A., Costa, L., & Castro, H. (2023). Collaborative planning in non-hierarchical networks. *Applied Sciences*, 13(14), 8347. <https://doi.org/10.3390/app13148347>
- Day, D., et al. (2017). Improving throughput and latency of D-Bus. In *Proceedings of the International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS)* (pp. 809–812).
- Faratin, P., Sierra, C., & Jennings, N. R. (1998). Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems*, 24(3–4), 159–182. [https://doi.org/10.1016/S0921-8890\(98\)00029-3](https://doi.org/10.1016/S0921-8890(98)00029-3)
- Fatima, S., Kraus, S., & Wooldridge, M. (2014). *Principles of automated negotiation* (1st ed.). Cambridge University Press.
- Hasan, M., Saifullah, M. K., Kamal, M. A. S., & Yamada, K. (2024). Distributed broadcast control of multi-agent systems using hierarchical coordination. *Biomimetics*, 9(7), 407. <https://doi.org/10.3390/biomimetics9070407>
- Huang, Z., et al. (2025). Adaptive orchestration for high-frequency multi-agent negotiations. *Autonomous Agents and Multi-Agent Systems*, 39(1), 112–138.

- Jelasity, M., Montresor, A., & Babaoglu, O. (2005). Gossip-based aggregation in large dynamic networks. *ACM Transactions on Computer Systems*, 23(3), 219–252. <https://doi.org/10.1145/1082469.1082470>
- Jennings, N. R., Parsons, S., Noriega, P., & Sierra, C. (2001). Automated negotiation: Prospects, methods and challenges. *Group Decision and Negotiation*, 10(2), 199–215. <https://doi.org/10.1023/A:1008746126376>
- Khan, K., Ahmed, S., Rahman, T., Ali, M., & Hussain, F. (2025). A multiagent framework coordinating one-to-many concurrent composite negotiations. *IEEE Open Journal of Industry Applications*, 6, 717–727.
- Krawczyk, H., Bellare, M., & Canetti, R. (1997). HMAC: Keyed-hashing for message authentication (RFC 2104). Internet Engineering Task Force. <https://doi.org/10.17487/RFC2104>
- Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)* (pp. 413–422). <https://doi.org/10.1109/ICDM.2008.17>
- Liu, T., Wang, J., & Cheng, D. (2019). Game theoretic control of multiagent systems. *SIAM Journal on Control and Optimization*, 57(3), 1691–1709. <https://doi.org/10.1137/17M1137931>
- Moore, D. (2025). A taxonomy of hierarchical multi-agent systems. *arXiv Preprint*. <https://arxiv.org/abs/2508.12683>
- Nisan, N., Roughgarden, T., Tardos, É., & Vazirani, V. V. (Eds.). (2007). *Algorithmic game theory*. Cambridge University Press.
- Ongaro, D., & Ousterhout, J. (2014). In search of an understandable consensus algorithm. In *Proceedings of the USENIX Annual Technical Conference (USENIX ATC)* (pp. 305–319).
- Pinto, T., Fotouhi Ghazvini, M. A., Soares, J., Faia, R., Corchado, J. M., Castro, R., & Vale, Z. (2018). Decision support for negotiations among microgrids. *Energies*, 11(10), 2526. <https://doi.org/10.3390/en1102526>
- Ravindrakumar. (2023). AI-driven threat detection in distributed cloud systems. *ShodhKosh Journal*, 4(2). <https://shodhkosh.com/>
- Renting, B. M., Hoos, H. H., & Jonker, C. M. (2020). Automated configuration of negotiation strategies. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)* (pp. 1116–1124).
- Rose, S., Borchert, O., Mitchell, S., & Connelly, S. (2020). *Zero trust architecture (NIST Special Publication 800-207)*. National Institute of Standards and Technology. <https://doi.org/10.6028/NIST.SP.800-207>
- Rubinstein, A. (1982). Perfect equilibrium in a bargaining model. *Econometrica*, 50(1), 97–109. <https://doi.org/10.2307/1912531>
- Rustogi, S. K. (1999). *Empirical studies of coordination in decentralized multiagent systems* (Doctoral dissertation, University of Massachusetts Amherst).
- Salama, R., & Al-Turjman, F. (2025). Addressing cybersecurity vulnerabilities with cloud security. *NEU Journal for Artificial Intelligence and Internet of Things*, 4, 86–95.
- Sim, K. M. (2004). Negotiation agents that make prudent compromises. *Computational Intelligence*, 20(4), 643–662. <https://doi.org/10.1111/j.0824-7935.2004.00265.x>
- Sim, K. M., Guo, H., Shi, B., & Sun, L. (2008). Negotiation and scheduling mechanisms for multi-agent systems. *Multiagent and Grid Systems*, 4(1), 1–3.
- Stephen, S. (2026). Federated learning architectures for distributed multi-agent systems. *arXiv Preprint*. <https://arxiv.org/>
- Suggu, S. K. (2025). Agentic AI workflows in cybersecurity. *Journal of Information Systems Engineering and Management*, 10(52s), 612–624.
- Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction* (2nd ed.). MIT Press.
- Tupsamudre, H., Kumar, A., Agarwal, V., Gupta, N., & Mondal, S. (2022). AI-assisted controls change management for cloud cybersecurity. In *Proceedings of the AAAI Conference on Artificial Intelligence* (pp. 12629–12635). <https://doi.org/10.1609/aaai.v36i11.21510>
- Tykhonov, D. (2010). *Designing generic and efficient negotiation strategies*. Data Archiving and Networked Services (DANS).
- Wooldridge, M. (2009). *An introduction to multiagent systems* (2nd ed.). Wiley.
- Xu, X., & Zhao, Q. (2020). Distributed no-regret learning in multiagent systems. *IEEE Signal Processing Magazine*, 37(3), 84–91. <https://doi.org/10.1109/MSP.2020.2970358>

-
- Xu, Y., Wang, L., Chen, X., & Zhao, Q. (2017). Cost-efficient negotiation over multiple resources with reinforcement learning. In *Proceedings of IEEE International Workshop on Quality of Service (IWQoS)*. <https://doi.org/10.1109/IWQoS.2017.7969152>
- Yamanaka, H., et al. (2022). Design and implementation of an edge computing testbed. *IEICE Transactions on Information and Systems*, *E105.D*(9), 1516–1528. <https://doi.org/10.1587/transinf.2021EDP7208>
- Yang, J., Li, H., Zhang, W., Chen, Y., & Zhao, P. (2025). Ablation analysis of centralized AI-driven negotiation systems. *IEEE Transactions on Autonomous Systems*, *12*(3), 204–219.
- Ygge, F., & Akkermans, H. (1999). Decentralized markets versus central control: A comparative study. *Journal of Artificial Intelligence Research*, *11*, 301–333. <https://doi.org/10.1613/jair.630>