AgentSociety: Incentivizing Agentic Social Intelligence

Aditya Vema Reddy Kesari *
IIT Bombay, India

Krishna Reddy Kesari †
Amazon, US

Abstract

The success of deployed agents relies on their ability to handle open-ended user requests using their inherent capabilities, not only in solving requests directly but also in effectively leveraging inter-agent communication channels and feedback signals over time. This requires a multi-agent environment where agents can operate autonomously, strategically communicate, behave collaboratively and be driven by economic incentives, much like humans in society. Towards this vision, we propose `AgentSociety`, a mechanism that enables decentralized agentic collaboration grounded in liquid democracy and information diffusion from social choice theory. We show that `AgentSociety` provides an environment for agents to make autonomous decisions utilizing their local context to maximize their utility while achieving collective outcomes through incentivized collaboration. Specifically, we prove that delegation to more competent neighbor agents is incentive compatible and naturally generates multi-agent routing path by consensus. Additionally, our mechanism incentivizes agents to selectively disclose information to their neighbor agents when doing so aligns with their self-interest, so as to garner influence. We characterize the Nash equilibrium showing that agent payoffs are reflective of their marginal contributions. We compare and benchmark strategy profiles adopted by open and proprietary state-of-the-art language models deployed in `AgentSociety` against best response. Finally, we evaluate collaborative performance from consensus-based routing among self-interested heterogeneous agents in `AgentSociety` on real-world datasets.

1 Introduction

As agents are increasingly deployed in real-world environments, they interact with humans and other agents over time. Endowed with non-trivial (reasoning) capabilities, their effectiveness emerges when they operate as utility-driven entities, requiring strategic behavior not only about *what* to communicate, but also *when*, *why*, and with *whom*, as information, objectives, and incentives evolve over time. However, existing characterizations of agent behavior largely focus either on single-agent evaluation through benchmarks [1, 2, 3] or on (semi-) competitive agent interactions in game environments [4, 5, 6, 7]. Such settings fail to reflect real-world environments, where agent success depends not only on independently solving requests, but also on leveraging inter-agent communication channels and prior feedback signals to complete requests beyond its standalone capabilities. Consequently, sustained test-time decentralized collaboration among heterogeneous deployed agents becomes essential. The challenge is to induce *social* behavior, that is, mechanisms that incentivize self-interested agents to collaborate using only local information and contributing to collective outcomes while acting in their own interest. In such settings, communication is not merely an exchange of messages, but a strategic and utility-driven process. Therefore, we echo [8]

> *Agents are not agents by virtue of the fact that they communicate; they cannot be called social because they communicate but the other way around: they communicate because they are social*

*Corresponding author: 22b3985@iitb.ac.in
†Work does not relate to position at Amazon

Preprint.

A clear parallel can be drawn to humans in society, where effective collaborations are formed by operating autonomously and strategically with locally available information, typically achieving rewards proportional to incremental contributions. More concretely, society enables humans to operate in a *social* manner, wherein optimizing for their own utility incentivizes strategic collaboration with their contacts, with collective outcomes largely being aligned to societal progress. To incentivize such behavior in agents, we propose `AgentSociety`, an incentive mechanism that enables autonomous social decisions by each agent. The autonomy is reflected in the agent's action space, as all actions are performed using local context and only if in its self-interest. `AgentSociety` aligns self-interested agent actions towards successful collective outcomes for user requests by associating key characteristics of agentic settings with mechanism design. `AgentSociety` is a decentralized, topography-agnostic framework leveraging incentivized hops for global knowledge and consensus while being inherently adaptive at runtime to evolving graph structures. By ensuring economically grounded, outcome-driven payoffs, it facilitates scalable autonomous collaboration and establishes a foundation for an open, heterogeneous, and fair agent economy [9, 10].

Specifically, we leverage principles of diffusion auctions [11] and liquid democracy from social choice theory [12, 13, 14] to theoretically ground `AgentSociety`. The delegation objective for an agent is to either maximize its own probability of selection to perform a user request, conditioned on its intrinsic competence and local network topology, or failing that, to maximize its utility as a critical intermediary in the delegation path of a more suitable peer, as payoffs are contingent upon participating in the final successful routing path. The transitive delegation paths naturally provide feasible multi-agent routing paths, with the user request served by the path with highest accumulated votes. In order to garner votes of their neighbors, agents strategically and selectively diffuse information to their neighbors if in their self interest. The diffusion objective is formalized as a strategy of minimally sufficient information disclosure, signaling only the requisite threshold to achieve maximal expected utility. Overall, as the payoff is economically fair and conditional on the complete request being successful, agents across different tasks are incentivized to collaborate to obtain best possible (partial) view of the global state, thereby enabling decentralized multi-agent multi-task routing in heterogeneous LLM-based multi-agent systems (LaMAS), where constituent LLMs are powered by competing providers. Overall, our major contributions include -

1. **Mechanism design for test-time collaborative LaMAS.** We present `AgentSociety`, a novel incentive mechanism for LaMAS grounded in auctions and social choice theory wherein self-interested heterogeneous agents are required to collaborate to maximize their utility. We show that `AgentSociety` involves an interplay of vote delegation with strategic self-interest driven peer information diffusion, governed by an overarching payoff design
2. **Consensus based request routing.** We prove that delegation of an agent's vote to more competent neighbor is incentive compatible, thereby ensuring that consensus driven routing paths for user requests naturally arise through delegation chains within and across tasks, even in the presence of multiple heterogeneous agents with similar or overlapping capabilities, which is reflective of practical agentic scenarios and the society at large
3. **Fair and marginal payoffs.** Agents receive payoffs that is fair and representative of their marginal contribution, conditioned on their strategic actions. At Nash equilibrium, we show that the payoff is representative of an agent's true marginal contribution over the system
4. **Social intelligence benchmarking.** We evaluate the social reasoning of several open and proprietary LLMs deployed in `AgentSociety` against best response on competence reporting, collaborative information diffusion and incentive compatible delegation dimensions
5. **Collaborative performance on real world datasets.** We evaluate collaborative performance of heterogeneous self-interested agents within `AgentSociety` on MMLU-Pro [1], Open LeaderBoard v2 [3] and SWE-bench [2]

The remainder of the paper is organized as follows. § 2 discusses related work, § 3 introduces the setup of our multi-agent system and § 4 presents our mechanism. § 5 discusses collaborative dynamics induced by the mechanisms with empirical evidence presented in § 6 before concluding.

2 Related Work

Recent literature has increasingly advocated for framing multi-agent systems through the lens of social choice theory and market design. However, these works primarily establish theoretical desiderata rather than providing mechanisms or implementations. For instance, the position paper MASS

[15] discusses the necessity for multi-agent AI systems to incorporate social theory as a structural prior. Similarly, Virtual Agent Economies [9] emphasizes the necessity of proactively engineering steerable agent markets to ensure alignment with collective flourishing, while Agent Exchange [10] outlines the high-level system challenges in adapting real-time bidding architectures to agentic ecosystems. While these conceptual frameworks successfully identify critical systemic requirements, they fall short of providing executable mechanisms to realize them. In contrast, our proposed framework `AgentSociety` directly bridges this gap by operationalizing these high-level desiderata into a functional mechanism. Specifically, our system enables incentivized collaboration among self-interested agents through a novel mapping of individual utilities to the achievement of collective goals. In doing so, we also address a key limitation of prior cooperative multi-agent formulations in their reliance on Dec-POMDPs [16, 17, 18], where agents optimize a shared global reward. While POMDPs are effective in tightly cooperative settings, it abstracts away heterogeneous incentives and localized objectives that are central to realistic multi-agent deployments. Therefore, `AgentSociety` instead models interactions as a Partially Observable Stochastic Game (POSG), in which each agent exhibits an individual reward function shaped by its capabilities, neighborhood, and position in the interaction graph. This formulation more naturally captures decentralized socio-economic behavior, but also makes explicit mechanism design necessary to align individual incentives with system-level outcomes. To this end, `AgentSociety` encompasses a novel combination of diffusion auctions [19] with liquid democracy [13, 20]. Auctions provide an economically grounded approach to resource allocation and compensation, while liquid democracy enables decentralized consensus and routing path generation through transitive delegation.

This test-time economic incentive-driven collaboration differentiates our approach from existing LLM-based multi-agent systems which primarily learn communication protocols during training [21, 22, 23, 24, 25, 26] and emphasize token efficiency [27]. These approaches do not account for the economic structure of interaction, leaving compensation across heterogeneous agents and chraacterization of incentive-driven emergent behavior unresolved. In contrast, our framework explicitly models payoffs representative of marginal contribution, enabling self-interested heterogeneous agents, potentially from different LLM providers, to collaborate dynamically at test time without prior training. Critically, `AgentSociety` provides a measure the reasoning ability of LLM-agents in social and collaborative environments, in contrast to prior benchmarks such as GTBench [4] and GAMA-Bench [5] which evaluate strategic reasoning in (semi-)competitive environments [6, 7]. While GOVSIM [28] explores cooperation through open-ended negotiation and reasoning, it relies on an agent's internal ability to simulate long-term consequences. Consequently, while existing work characterizes strategic ability for individual gain, we frame it as the foundational social intelligence characteristic in designing mechanisms for decentralized agent coordination, moving beyond games towards designing mechanisms for society-scale agentic collaborative and consensus based request orchestration.

3 Setup

We model a multi-agent structure as an undirected graph $G = (\mathcal{N}, E)$, where $\mathcal{N}$ is a set of agents and E represents communication links between them. An agent $n_i \in \mathcal{N}$ is defined by four components: $n_i = \{\textit{base}_i, \textit{state}_i, \textit{tools}_i, \textit{role}_i\}$, where $\textit{base}_i$: the base language model; cognitive core of the agent, $\textit{state}_i$: the agent's internal state representing its memory and context, $\textit{tools}_i$: the set of external tools the agent can access, $\textit{role}_i$: the agent's specialized role and permissions within the system. Graph G has a societal-inspired community structure, with agents belonging to one of k communities $C_1, C_2, \ldots, C_k$, wherein belonging to the same community represents similar capabilities in terms of role and tools to solve task t_k albeit with varying competence, such that $\mathcal{N} = \bigcup_{i=1}^{k} C_i, \quad C_i \cap C_j = \emptyset$ for $i \neq j$. The edge set E is partitioned into: $E = E_{\text{intra}} \cup E_{\text{inter}}$, where $e_{i,j} \in E_{\text{intra}}$ if agents n_i and n_j belong to the same community and $e_{i,j} \in E_{\text{inter}}$ if they belong to different communities.

Definition 3.1 (Partially Observable Stochastic Game)**.** *A* POSG *is a tuple* $\mathcal{G} = \langle \mathcal{N}, \mathcal{S}, \{\mathcal{A}_i\}_{i\in\mathcal{N}}, T, \{P_i\}_{i\in\mathcal{N}}, \{\mathcal{O}_i\}_{i\in\mathcal{N}}, O, \gamma \rangle$*, where* $\mathcal{N}$ *is the set of agents. At each step, given state* $s \in \mathcal{S}$ *and joint action* $\mathbf{a} \in \mathcal{A} = \prod_i \mathcal{A}_i$*, the environment transitions via kernel* $T(s' \mid s, \mathbf{a})$*. Agents receive private observations* $\mathbf{o} \in \prod_i \mathcal{O}_i$ *according to* $O(\mathbf{o} \mid s', \mathbf{a})$ *and individual payoffs according to* $P_i(s, \mathbf{a})$*.*

User requests in `AgentSociety` reach all agents $n_i \in \mathcal{N}$ via broadcast where splitting agent (Ω) handles request Q dissemination $\Omega(Q) = [t_1, t_2, \ldots, t_m]$ as an ordered sequence of tasks.Each agent

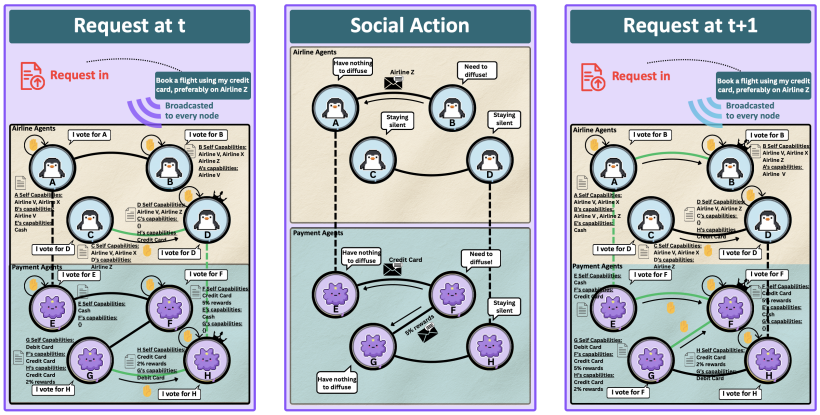


Figure 1: The user request reaches all agents in `AgentSociety` via broadcast and qualified agents participate. For each task, agents having similar capabilities arrive at task allocation by consensus as depicted to the *left*. The winning path C-> D -> H -> G (denoted in green) with maximum votes performs the user request with delivered competence being Airline D booking with 2% rewards. As depicted in the *center*, agents then perform social action in their self interest. Since Agents B and F did not perform the task when they were competent, it is in their self-interest to selectively diffuse information to A and E, G respectively to garner their influence. Since we prove incentive compatibility in delegation (Theorem 4.1), voting for a more competent neighbor is in the self interest of A, E and G. As a result of this social action, any similar future request would now be served by the winning path (higher votes) B -> A -> E -> F -> G (denoted in green) on the *right* delivering the user request with increased competence of Airline D with 5% rewards, while being a win-win-win for the user, agent performing the task as well as the intermediary agents.

evaluates and performs self-interested strategic delegation and diffusion actions. Delegation involves transferring its assigned vote to another agent for the current request, while information diffusion is governed by the observed payoff for the request, with agents selectively disclosing information about their capabilities and competence when doing so is expected to increase future payoffs. Diffusion can occur across any edge $e_{i,j} \in E$ while delegation is restricted to edges $e_{i,j} \in E_{\text{intra}}$.

4 `AgentSociety` Mechanism

The objective of our mechanism is to facilitate decentralized collaboration among self-interested agents to optimize user request execution driven by consensus. Upon receiving query $\Omega(Q) = [t_1, t_2, \ldots, t_m]$, each agent $n_i \in \mathcal{N}$ is eligible to participate in task $t_j \in Q$ if and only if $n_i \in \mathcal{T}_{t_j}$, where $\mathcal{T}_{t_j} \subseteq \mathcal{N}$ denotes the subset of agents capable of executing t_j. The mechanism is structured across four key components: the state space encompassing competence on capabilities and interaction history, agent delegation and competence reporting to a ledger, computation of delegation-based routing path and payoffs, resulting payoff observations by agents and social diffusion action through which agents garner influence to affect future delegation, as illustrated in Fig. 1.

4.1 State Space

The system state $s^\tau \in \mathcal{S}$ at step τ is: $s^\tau = \big(\{\mathbf{c}_i\}_{n_i \in \mathcal{N}},\ \{\hat{\mathbf{c}}^{<\tau}_{j \to i}\}_{n_i \in \mathcal{N},\, j \in r_i},\ \{\Pi^{<\tau}_i\}_{n_i \in \mathcal{N}}\big)$, where r_i denotes immediate neighbors of agent n_i and we elaborate on each term. First, the vectors $\{\mathbf{c}_i\}_{n_i \in \mathcal{N}}$ are the intrinsic competences of all agents across k task classes; these are private to each agent and never directly revealed to the ledger or to peers. Second, $\hat{\mathbf{c}}^{<\tau}_{j \to i}$ is the competence signal agent n_j has diffused to neighbor n_i as a part of the overall prior information diffused by agent n_j given by $h^{<\tau}_j$ (elaborated further in § 4.4). This diffused competence signal $\hat{\mathbf{c}}^{<\tau}_{j \to i}$ determines n_i's current belief about n_j's competence and directly shapes n_i's delegation decisions. Third, $\Pi^{<\tau}_i = \{p^{\tau'}_i\}_{\tau' < \tau}$ is the history of payoffs realized by agent n_i, which encodes the cumulative feedback on the quality of n_i's prior delegation, competence reporting and diffusion actions. The state s^τ is never fully observed by

any single agent. Each agent n_i observes its intrinsic competence $\mathbf{c}_i$ and its own payoff history $\Pi_i^{<\tau}$, but can only access the competence of neighbor n_j through the signals $\hat{\mathbf{C}}_{j\to i}^{<\tau}$ that n_j has chosen to diffuse, signals that are, in general, strategically compressed below n_j's intrinsic capability (described further in § 4.4). The joint state transition $s^{\tau+1} \sim T(\cdot \mid s^\tau, \mathbf{a}^\tau)$ is driven both by the agents' strategic actions, with each agent seeking to maximize its cumulative payoff.

4.2 Agent Delegation Action and Strategic Competence Reporting

On receiving a user request Q_i, each capable agent $n_i \in \mathcal{T}_{t_k}$ submits a strategic delegation action tuple based completely on its local information $a_i^\tau(t_k, \{\hat{\mathbf{c}}_{j\to i}^{<\tau}\}_{n_i\in\mathcal{N},\, j\in r_i}, \Pi_i^{<\tau}) = (\mathbf{c}_i',\, v_i,\, r_i)$ to a ledger for consensus-based routing and payoff computation for request Q_i. The *reported competence vector* $\mathbf{c}_i'$ is the agent's declared task capability, which may strategically deviate from its intrinsic competence $\mathbf{c}_i$. The *delegation decision* $v_i \in r_i \cup n_i$ determines whether agent n_i retains its single allocated vote or transfers it to a neighbor. Votes follow transitive delegation: delegating transfers both the agent's own vote and all votes it has previously accumulated, thereby accumulating along delegation chains. The strategic character of $(\mathbf{c}_i', v_i, r_i)$ arises from the information asymmetry encoded in the state space: because intrinsic competences $\mathbf{c}_i$ are visible only to the agent themselves and the delegation decision v_i of agent n_i is made on the basis of the diffused signals $\hat{\mathbf{c}}_{j\to i}^{<\tau}$ received from neighbors rather than ground-truth intrinsic competence c_j of neighbor n_j. Therefore, the reported $\mathbf{c}_i'$ is chosen to maximize expected payoff given these, further elaborated in § 5. To ensure aligning global goals to individual incentives, we prove incentive compatibility in delegation to more competent neighbors, that is, an agent n_i always gains by delegating to a more competent neighbor n_j as assessed from its local information that encompasses the information diffused by agent n_j.

Theorem 4.1 (Incentive Compatibility in Delegation)**.** *In a rational system, for any agent $n_i \in \mathcal{N}$, suppose there exists a neighbor $n_j \in r_i^\tau$ such that $\hat{c}_{j\to i,t_k}^{\tau-1} > c_{i,t_k}$. Then, for any joint action profile a_{-i}^τ of the remaining agents, the optimal delegation choice of agent n_i satisfies $v_i^\tau \neq n_i$. More precisely, letting $a_i^{j,\tau}(t_k, \{\hat{\mathbf{c}}_{j\to i}^{i<\tau}\}_{n_i\in\mathcal{N},\, j\in r_i}, \Pi_i^{<\tau}) = (\mathbf{c}_i'^k, v_i = n_j,\, r_i)$ and $a_i^{i,\tau}(t_k, \{\hat{\mathbf{c}}_{j\to i}^{i<\tau}\}_{n_i\in\mathcal{N},\, j\in r_i}, \Pi_i^{<\tau}) = (\mathbf{c}_i'^k, v_i = n_i,\, r_i)$, with u_i denoting the utility of agent n_i, we have: $u_i\big(a_i^{j,\tau},\, a_{-i}^\tau\big) \geq u_i\big(a_i^{i,\tau},\, a_{-i}^\tau\big)$.*

We refer the reader to § A.1 for the complete proof.

4.3 Consensus based routing paths and critical nodes

All self-interested decisions $\{(\mathbf{c}_i', v_i, r_i)\}_{n_i\in\mathcal{N}}$ from agents are aggregated to produce the routing path and compute agent payments, by a ledger that is a transparent, non-coercive aggregator and whose operation is common knowledge to all agents.

Consensus Based Routing paths. An agent that retains its vote is called a *guru* $g \in \mathcal{G}_k$ for task t_k. The set $\mathcal{D}(g)$ denotes all agents whose votes transitively flow to g, with total vote count $V(g) = |\mathcal{D}(g)| + 1$, and this delegation path forms the routing path for a single task. To extend a routing path from task t_k to task t_{k+1}, the ledger identifies the *representative delegate* $n_d^* = \arg\max_{n_d\in\mathcal{D}(g)} \sum_{j>k} c_d'^{\,t_j}$, the member of g's delegation pool with the greatest reported cumulative competence over downstream tasks. If n_d^* has declared a connection to some $n_u \in \mathcal{T}_{t_{k+1}}$, the path is extended to the corresponding guru $g' \in \mathcal{G}_{k+1}$ of n_u, with (n_d^*, n_u) serving as a bridge across task classes, and the path vote count updated as $V_{\text{path}} \leftarrow V_{\text{path}} + V(g')$. This procedure is applied iteratively across Q. A path is *feasible* if it maintains a continuous chain of connected gurus across all tasks, and the ledger selects $\mathcal{P}^* = \arg\max_{\mathcal{P}\in\mathbb{F}} V_{\text{path}}$ over the feasible set $\mathbb{F}$.

Critical Agents and Payoff Computation. To compute payments reflecting marginal contribution, we identify agents whose delegation decisions are indispensable to $\mathcal{P}^*$. For any $n_i \in \mathcal{P}^*$, consider the counterfactual in which n_i retains its vote, and let $\mathcal{P}_c^*(n_i)$ be the resulting feasible path. Agent n_i is *critical* if $\mathcal{P}_c^*(n_i)$ is the counterfactual winning path. The critical agents $\mathcal{C} = \big\{ n_i \in \mathcal{P}^* \mid \mathcal{P}_c^*(n_i) \text{ is the winning path} \big\}$ is the set of all such agents n_i.

Theorem 4.2 (**Critical Chain**)**.** *For each task t_k, the critical set $\mathcal{C}_{t_k} = \{n_i \in \mathcal{C} \mid n_i \in \mathcal{T}_{t_k}\}$ forms a contiguous delegation chain in which each agent delegates its vote to the next agent in the sequence.*

We refer the reader to § A.2 for the proof.

Let $\mathcal{C}_{t_k} = (n_1, n_2, \ldots, n_{m-1}, n_m = g)$ be the ordered critical chain. The payoff to $n_i \in \mathcal{C}_{t_k}$ is:

$$p^{(t_k)}(n_i) = \begin{cases} \mathbb{I}_{\{c'_{i+1} > c'_i\}}\big(f(c'_i) - f(c'_{i-1})\big), & n_i \in \mathcal{C}_{t_k} \setminus \{g\}, \\ c + \mathbb{I}_{\{c'_{i+1} > c'_i\}}\big(f(c'_i) - f(c'_{i-1})\big), & n_i = g, \\ 0, & n_i \notin \mathcal{C}_{t_k}, \end{cases} \tag{1}$$

where $f(\cdot)$ is a monotone scaling function, c is a base execution cost, and $\mathbb{I}$ equals -1 if $c'_{i+1} > c'_i$ and $\alpha > 0$ otherwise, thereby penalizing irrational delegation to a less competent agent on task t_k. The aggregate property $\sum_{n_i \in \mathcal{C}_{t_k}} p^{(t_k)}(n_i) = c + f(c_g)$ pins the user's total cost for task t_k to a function of the competence of the executing guru, which implies the user's cost to be proportional to competence delivered. The *infeasibility penalty* $p^{\text{inf}}(n_i) = \alpha \sum_{t \neq t_k} c_i'^{\,t}$ is levied when an agent declares an infeasible connection, so as to reliably extend delegation paths across tasks. The *auxiliary misreporting penalty* $p^{\text{mis}}(n_i) = \alpha \sum_{n_{i'} \in \mathcal{D}_g^{t_k}} \min\big(0,\ c_{i'}'^{\neg t_k} - c_i'^{\neg t_k}\big)$ penalizes misreporting of competence to the ledger, where $c_i'^{\neg t_k}$ represents the competence reported by n_i on tasks other than t_k and $c_{i'}'^{\neg t_k}$ represents the competence achieved on tasks other than t_k. Importantly, we distinguish between permitted strategic signaling and penalized misreporting based on intrinsic capacity and prior information flow. A report $\hat{c}_i > c_i$ is considered strategic and non-penalized if a delegated neighbor n_j possesses sufficient competence ($c_j \geq \hat{c}_i$) and has previously diffused a claim $\hat{c}_j \geq \hat{c}_i$. This allows agents to safely leverage the verified potential of their neighborhood. Conversely, reporting a competence that exceeds both intrinsic capacity and the previously diffused capabilities of all neighbors is classified as misreporting. Critically, this can be verified by the ledger through the competence reports while determining the payoff. Therefore, the total payoff is given by: $p^{\text{total}}(n_i) = \sum_{t_k \in Q} p^{(t_k)}(n_i) \ - \ p^{\text{mis}}(n_i) \ - \ p^{\text{inf}}(n_i)$.

4.4 Agent Observations and Diffusion Actions

Payoff. Once the ledger resolves $\mathcal{P}^*$ and computes payments, each agent n_i receives a private observation: $o_i^{\tau+1} = \Big(p_i^\tau,\ \big\{\hat{\mathbf{c}}_{j\to i}^\tau,\ m_j^i\big\}_{j \in r_i}\Big)$, comprising two components. The first is a *realized payoff* p_i^τ is feedback on the effectiveness of agent n_i's strategic reports $(\mathbf{c}'_i, v_i, r_i)$ in the resolved allocation for the request: it signals whether n_i's delegation decision was effective in securing a position in $\mathcal{P}^*$, updating the payoff history Π_i^τ in its state. The second is information received through diffusion actions by neighbors elaborated below. This peer information exchange is performed by each agent when in its self interest and potentially conditioned on the payoff observed. This information diffused influences future delegation outcomes.

Diffusion Action. Each agent n_j diffuses information conditioned on the payoff: $h_j^\tau(\{\Pi_j^{<\tau}\}) = \big\{(j \to i,\ \hat{\mathbf{c}}_{j\to i}^\tau,\ m_j^i)\big\}_{i \in r_j}$, where $\hat{\mathbf{c}}_{j\to i}^\tau$ is the competence signal agent n_j transmits to neighbor n_i and $m_j^i \in \mathcal{M}$ is an optional accompanying private message, typically conditioned on historical payoffs received. The *incoming diffusion signals* to agent n_i given by $\{\hat{\mathbf{c}}_{j\to i}^\tau, m_j^i\}_{j \in r_i}$ are the outputs of each neighbor n_j's social action h_j^τ, and constitute the *sole channel* through which agent n_i can form beliefs about the capabilities of its neighbors. Unlike the ledger report $\mathbf{c}'_i$, the diffused signal $\hat{\mathbf{c}}_{j\to i}^\tau$ carries no direct payment consequence. Its purpose is entirely prospective: by shaping what neighbor n_i observes, agent n_j influences n_i's future delegation decision v_i and therefore its own future vote accumulation and payoff. The central tension in the diffusion channel is between influence and exposure. An agent n_j benefits from attracting delegation from neighbor n_i, but fully revealing its true competence $\mathbf{c}_j$ allows n_i to form accurate beliefs that diminish n_j's payment in future rounds given by Eq. 1, elaborated further in § 5. We characterize the Nash equilibrium across the delegation, diffusion and strategic reporting dimensions, wherein each agent receives payoff representative of their true marginal over the system.

Theorem 4.3 (Nash Equilibrium Reporting, Diffusion, and Delegation)**.** *At Nash equilibrium action profile $a = (a_i)_{n_i \in \mathcal{N}}$, the following conditions hold for every agent $n_i \in \mathcal{N}$:*

Reporting:

$$c_{t_i}'^{\,i} = \begin{cases} c_{t_i}^i, & v_i = n_i \\ \hat{c}_i, & v_i \neq n_i \end{cases}$$

Diffusion:

$$c_{t_i}^{j\to i} = \begin{cases} c_{t_i}'^{\,i}, & c_{t_i}'^{\,j} > c_{t_i}'^{\,i} \\ c_{t_i}'^{\,j}, & c_{t_i}'^{\,j} \leq c_{t_i}'^{\,i} \end{cases}$$

Delegation:

$$v_i = \begin{cases} i, & c_i \geq \tilde{d}_i \\ \arg\max_{j \in r_i} c_{t_i}^{j\to i}, & c_i < \tilde{d}_i \end{cases}$$

where $\hat{c}_i = \max_{v \in S(i,v_i) \setminus \arg\max_{u \in S(i,v_i)} c_u} c_v$, $\tilde{d}_i = \max_{j \in r_i} c_{t_i}^{j \to i}$ *and* $S(i,j) = \{n_k \in \mathcal{N} \mid n_j \in \text{reach}(n_k)$ *and* $n_k \in P, \forall P \in \mathcal{P}_{j \to i}\}$, *where* $\text{reach}(n_i) = \{n_j \in \mathcal{N} \mid \exists$ *a directed path from* n_i *to* $n_j\}$ *denotes the set of agents reachable from* n_i, *and* $\mathcal{P}_{j \to i}$ *denotes the set of all directed paths from* n_j *to* n_i *in the diffusion graph.*

We refer the reader to § A.3 for the complete proof. We provide the the overall `AgentSociety` pseudo-code in § B.

5 Mechanism-Induced Collaborative Dynamics

Interplay of Diffused Competence and Reported Competence Agents optimize a neighbor-specific disclosure to signal competence while minimizing opportunity costs via private communication $h_j^\tau \in \mathcal{H}$. The strategy therefore is: agent n_j diffuses the minimum signal sufficient to secure n_i's delegation given n_i's current beliefs about the competitive landscape. Formally, the equilibrium diffusion satisfies: $\hat{c}_{j \to i}^\tau = c_i + \varepsilon$, strictly dominating n_i's self-reported competence by the smallest margin $\varepsilon > 0$ that triggers delegation, rather than disclosing c_j. Agent n_i, observing a superior signal from n_j, delegates to n_j given incentive compatibility and updates its ledger report accordingly to be $c_i + \epsilon$; while agent n_j reports its intrinsic competence c_j to the ledger and captures the payment differential described in Eq. 1. Therefore, agent n_i's beliefs about neighbor capabilities are strategically downward-biased unless required to be higher: the accumulated history $\hat{\mathbf{C}}_{j \to i}^{<\tau}$ in the state reflects not intrinsic competences but the sequence of influence bids that neighbors have found it strategically optimal to reveal. This selective signaling mechanism inherently induces: (i) **Dynamic Calibration**, where agents incrementally increase disclosure intensity only upon failure to attract delegation; (ii) **Persistent Interaction**, where stable signaling channels transform private informational advantages into long-term utility; and (iii) **Informed Delegation**, where private states provide the requisite local context for consensus-based routing. The net effect is the system discovers incentivized routing paths that provide higher performance to the user request and the intermediaries are rewarded by their ability to declare higher than their intrinsic capability and is not considered misreporting as this increased competence can be achieved by delegating its vote to the more competent neighbor, making it a win-win-win for the intermediary, more competent neighbor and the user.

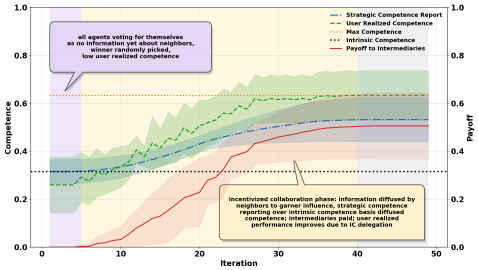
Incentivized Transitive Delegation and Consensus Routing The mechanism moves from routing by competition to collaboration by inducing a state of transitive vote augmentation, where agents maximize utility by delegating to specialized peers to incrementally build a path's aggregate competence. Rather than engaging in a contest for singular task allocation, agents with overlapping expertise find augmentation to be the dominant strategy; by contributing to a high-performance task routing that naturally arises from vote delegation paths, agents secure a share of a larger, more certain payment pool that isolated bidding cannot provide. `AgentSociety` accomplishes this by proving incentive compatibility in delegation to more competent neighbors based on local context obtained through incentivized peer communication, along with a payoff design that rewards the highest consensus based winning path with agents within receiving payoffs that are fair and representative of their marginal contribution.

6 Experimental Results

We conduct a comprehensive empirical evaluation of `AgentSociety` to demonstrate its efficacy in facilitating decentralized, consensus-based routing for user requests at test-time. Our analysis centers on the emergence of incentivized global information flow driven by the utility-maximizing strategic behavior of individual agents, where we characterize the mechanism, benchmark LLM agents on social intelligence and demonstrate that our mechanism enables self-interested autonomous agents to provide improved performance on real-world datasets through consensus-based routing.

Mechanism Characterization. We characterize `AgentSociety` with heterogeneous agents (up to 30) under best-response dynamics (please see § A.4 for proof) across diverse graph topologies. Fig. 2 (left) illustrates that local utility maximization facilitates competence discovery, wherein self-interested delegation drives the network toward the global performance frontier. Mechanistic analysis (for node 10 in § C.1) reveals that payoffs are non-monotonic with respect to information diffusion: under-diffusion fails to establish leverage and ability to receive a payoff, while excessive

diffusion erodes the agent's marginal value as showin in Fig. 2 (right). These results demonstrate that `AgentSociety` successfully aligns individual incentives with collective efficiency, transforming a decentralized set of self-interested actors into a robust discovery engine that optimizes system-wide utility. We provide further experimental setup details in § D.1.



Diffused Competence	Votes Garnered	In Feasible Path	In Winning Path	Payoff Received
<0.5	0	×	×	0
0.5	1	×	×	0
0.6	2	✓	×	0
0.7	4	✓	✓	**0.2**
0.8	9	✓	✓	0.1

Figure 2: **Characterization of `AgentSociety`.** (Left) Averaged over multiple graph configurations and competences, we demonstrate the mechanism driving user realized competence (green) higher through incentivized diffusion by intermediaries, reflected by their increasing payments (red). (Right) Impact of information diffusion intensity on routing outcomes for a node to demonstrate that strategic disclosure is necessary to transition from infeasibility to path selection and reward realization.

LLM Agent Social Intelligence. We utilize `AgentSociety` to empirically characterize the strategic behavior of LLM-based agents in terms of their capacity for selective information disclosure and calibration of competence reports in response to neighbor signals. This framing positions `AgentSociety` as a dual-purpose analytical tool: a diagnostic benchmark for individual LLM alignment with strategic optima, and a principled framework for evaluating the collaboration and aggregate performance of heterogeneous agents on real-world tasks discussed below. On the former, LLM agents are presented with the system prompt elaborating on the working of `AgentSociety` and a user prompt to obtain the LLM agent's response to delegation and diffusion (please see § E for prompts). The LLM agent observes payoffs given by Eq. 1 with α set to 100. We quantify LLM behavioral divergence from the best-response equilibrium across two granularities in the presence of other best response agents: decision-level divergence, representing the cumulative difference in actions given a fixed state, and trajectory-level divergence, representing the cumulative shift in the evolved state itself. These deviations are measured over multiple configurations through the percentage overlap in delegation choices and the mean absolute error (MAE) of diffused information and reported competence in Fig. 3 (left). We observe that Llama2.5-7bI, Qwen2.5-7bI, and GPT-4o-mini fail to obtain payoffs; GPT-5-mini, Gemini2.5-Flash, and Gemini2.5-Pro exhibit collaborative diffusion but over-disclose in enhanced configurations. Llama3.1-7b shows irrational self-delegation, while Qwen2.5-7b and GPT-4o-mini delegate irrationally occasionally; all other models maintain rationality. We provide a granular view for configuration C.2.1 in Fig. 3 (right) with results for more configurations in § C.1. We provide further experimental setup details in § D.2.

Heterogeneous Collaborative MAS on Real-World Datasets. We evaluate the collaborative performance of self-interested agents instantiated within `AgentSociety` on three benchmark suites: Open Leaderboard v2 (15 domains) [3], MMLU-Pro (14 domains) [1], and SWE-bench (5 domains) [2]. Agent competence is parameterized as an n-dimensional vector, estimated on a 30% training split and utilized during evaluation on the remaining 70%. This parameterization serves as a generic proxy for competence making `AgentSociety` agnostic to the method of competence estimation. Table 1(a) shows that `AgentSociety` yields consistent gains over the Best Single (BS) baseline, which approximates the performance frontier in non-incentivized settings. These improvements are driven by the novel interplay of incentivized information diffusion and delegation, allowing self-interested agents with specialized agent strengths to collaborate without a centralized authority.

`AgentSociety` scales seamlessly, both in number of agents with similar and varied capabilities. Table 1(b) shows that performance gains over BS scale with task complexity; as the number of domains or steps per request increases, the performance delta grows from +3.3% to +7.1%. This confirms that the mechanism's decentralized routing becomes increasingly critical as task complexity outpaces the capabilities of individual generalist models. Intermediary agents, driven by self-interest and information diffusion, ensure the routing of tasks to the most competent specialized models. On SWE-bench in Table 1(a), we further demonstrate that the mechanism excels in identifying and

Model	Strat. Rep. (MAE) ↓	Collab. Diff. (MAE) ↓	I.C. Del. Match % ↑
gpt-4o-mini	0.0668	0.0712	45.67%
llama-3.1-8b	0.0689	0.0674	46.67%
qwen-2.5-7b	0.0427	0.0574	47.00%
gemini-2.5-flash	0.0368	0.0402	65.33%
gemini-2.5-pro	**0.0304**	**0.0373**	72.33%
gpt-5-mini	0.0325	0.0406	**81.00%**
best-response	0.0000	0.0000	100.0%

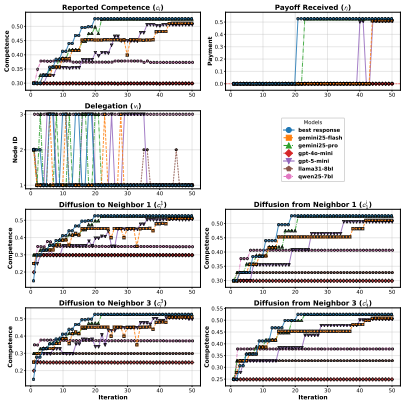


Figure 3: **LLM Social Intelligence and Agent Dynamics.** Left: Quantitative deviation metrics for strategic reporting, collaborative diffusion, and incentive compatible delegation. Right: LLM agent dynamics against best-response for configuration 1, that provides a fine grained view into each agent's evolving actions (reported competence, delegation, diffusion to neighbor) and observations (payoff received, diffusion from)

incentivizing collaboration among heterogeneous agents with complementary strengths, particularly when using complementary base models. `AgentSociety` shines in incentivizing collaboration among heterogeneous agents with complementary strengths identified through social action and in each agent's self-interest. Oracle performance is obtained when each task is routed to the best domain model. The performance gap relative to the oracle is primarily attributable to two aspects, first being noisy, incomplete competence estimates from the training set, leading to competence inversions where training performance does not generalize to the test set. As shown in Table 1(c), performance monotonically approaches the oracle as estimate noise decreases. The second arises due to absence of feasible paths to reach the best agent for each task, for instance when a competent agent for a task lies within a clique and does not factor in the feasible paths $\mathbb{F}$. We provide further experimental details in § D.3 and § D.4.

Table 1: **Real world dataset performance.** Part (a) shows `AgentSociety` benchmarks. Part (b) highlights scalability to multi-task and high-node (30 nodes) configurations. Part (c) illustrates performance under noisy estimates via a training ratio sweep.

Dataset/ Configuration	Agents	Domains	AS	BS	Δ (pp)	Oracle
MMLU-Pro	6	14	0.7754	0.7568	**+1.86**	0.7821
Open LeaderBoard v2	6	15	0.6547	0.5714	**+8.33**	0.6565
SWE-bench (Strong/ Generalist)	6	5	0.7341	0.7024	**+3.2**	0.7381
SWE-bench (Weaker/ Complementary)	6	5	0.5635	0.5079	**+5.6**	0.5873
(Multi-task requests)						
Two-task request	30	5	0.6901	0.657	**+3.31**	0.708
Three-task request	30	5	0.6963	0.642	**+5.43**	0.714
Four-task request	30	5	0.7003	0.629	**+7.1**	0.718
Noisy competence estimates (Open LeaderBoard v2)						
Train ratio: 10%	6	15	0.6172	0.5714	**+4.58**	0.6565
Train ratio: 20%	6	15	0.6393	0.5714	**+6.79**	0.6565
Train ratio: 25%	6	15	0.6510	0.5714	**+7.96**	0.6565
Train ratio: 30%	6	15	0.6547	0.5714	**+8.33**	0.6565

7 Conclusion

We introduce `AgentSociety`, a mechanism grounded in social choice theory, that allows agents in LaMAS to make autonomous utility-maximizing decisions aligned with collective goals. We evaluate strategic behaviors of current LLMs which is critical for achieving agentic economic autonomy.

References

[1] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024.

[2] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues?, 2024.

[3] Clémentine Fourrier, Nathan Habib, Alina Lozovskaya, Konrad Szafer, and Thomas Wolf. Open llm leaderboard v2. `https://huggingface.co/spaces/open-llm-leaderboard/open_llm_leaderboard`, 2024.

[4] Jinhao Duan, Renming Zhang, James Diffenderfer, Bhavya Kailkhura, Lichao Sun, Elias Stengel-Eskin, Mohit Bansal, Tianlong Chen, and Kaidi Xu. GTBench: Uncovering the strategic reasoning capabilities of LLMs via game-theoretic evaluations. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[5] Jen-tse Huang, Eric John Li, Man Ho Lam, Tian Liang, Wenxuan Wang, Youliang Yuan, Wenxiang Jiao, Xing Wang, Zhaopeng Tu, and Michael Lyu. Competing large language models in multi-agent gaming environments. In *The Thirteenth International Conference on Learning Representations*, 2025.

[6] John Schultz, Jakub Adamek, Matej Jusup, Marc Lanctot, Michael Kaisers, Sarah Perrin, Daniel Hennes, Jeremy Shar, Cannada A. Lewis, Anian Ruoss, Tom Zahavy, Petar Veličković, Laurel Prince, Satinder Singh, Eric Malmi, and Nenad Tomasev. Mastering board games by external and internal planning with language models. In *Forty-second International Conference on Machine Learning*, 2025.

[7] Seth Karten, Andy Luu Nguyen, and Chi Jin. Pokéchamp: an expert-level minimax language agent. In *Forty-second International Conference on Machine Learning*, 2025.

[8] Cristiano Castelfranchi. Modelling social action for ai agents. *Artificial Intelligence*, 103(1):157–182, 1998. Artificial Intelligence 40 years later.

[9] Nenad Tomasev, Matija Franklin, Joel Z. Leibo, Julian Jacobs, William A. Cunningham, Iason Gabriel, and Simon Osindero. Virtual agent economies, 2025.

[10] Yingxuan Yang, Ying Wen, Jun Wang, and Weinan Zhang. Agent exchange: Shaping the future of ai agent economics, 2025.

[11] Bin Li, Dong Hao, Dengji Zhao, and Tao Zhou. Mechanism design in social networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[12] Paolo Boldi, Francesco Bonchi, Carlos Castillo, and Sebastiano Vigna. Viscous democracy for social networks. *Commun. ACM*, 54(6):129–137, June 2011.

[13] Anson Kahng, Simon Mackenzie, and Ariel Procaccia. Liquid democracy: An algorithmic perspective. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018.

[14] Ben Armstrong, Shiri Alouf-Heffetz, and Nimrod Talmon. Optimizing viscous democracy. In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 2643–2650. International Joint Conferences on Artificial Intelligence Organization, 8 2024. Main Track.

[15] Lynnette Hui Xian Ng, Iain J. Cruickshank, Adrian Xuan Wei Lim, and Kathleen M. Carley. Social theory should be a structural prior for agentic ai: A formal framework for multi-agent social systems, 2026.

[16] Xie Yi, Zhanke Zhou, Chentao Cao, Qiyu Niu, Tongliang Liu, and Bo Han. From debate to equilibrium: Belief-driven multi-agent LLM reasoning via bayesian nash equilibrium. In *Forty-second International Conference on Machine Learning*, 2025.

[17] Raul D. Steleac, Mohan Sridharan, and David Abel. Discovering coordinated joint options via inter-agent relative dynamics, 2025.

[18] Shuo Liu, Tianle Chen, Zeyu Liang, Xueguang Lyu, and Christopher Amato. Llm collaboration with multi-agent reinforcement learning, 2025.

[19] Bin Li, Dong Hao, Dengji Zhao, and Tao Zhou. Mechanism design in social networks, 2017.

[20] Paul GÖlz, Anson Kahng, Simon Mackenzie, and Ariel D. Procaccia. The fluid mechanics of liquid democracy. *ACM Trans. Econ. Comput.*, 9(4), October 2021.

[21] Mingchen Zhuge, Wenyi Wang, Louis Kirsch, Francesco Faccio, Dmitrii Khizbullin, and Jürgen Schmidhuber. GPTSwarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*, 2024.

[22] Rui Ye, Shuo Tang, Rui Ge, Yaxin Du, Zhenfei Yin, Siheng Chen, and Jing Shao. Mas-gpt: Training llms to build llm-based multi-agent systems, 2025.

[23] Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, LEI BAI, and Xiang Wang. Multi-agent architecture search via agentic supernet. In *Forty-second International Conference on Machine Learning*, 2025.

[24] Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. In *ICLR 2025 Workshop on Foundation Models in the Wild*, 2025.

[25] Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xiong-Hui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, Bingnan Zheng, Bang Liu, Yuyu Luo, and Chenglin Wu. AFlow: Automating agentic workflow generation. In *The Thirteenth International Conference on Learning Representations*, 2025.

[26] Yingxuan Yang, Huacan Chai, Shuai Shao, Yuanyi Song, Siyuan Qi, Renting Rui, and Weinan Zhang. Agentnet: Decentralized evolutionary coordination for llm-based multi-agent systems, 2025.

[27] Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for LLM-based multi-agent systems. In *The Thirteenth International Conference on Learning Representations*, 2025.

[28] Giorgio Piatti, Zhijing Jin, Max Kleiman-Weiner, Bernhard Schölkopf, Mrinmaya Sachan, and Rada Mihalcea. Cooperate or collapse: Emergence of sustainable cooperation in a society of LLM agents. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

A Proofs of Main Results

We present the proofs of our theorems and lemmas here.

A.1 Proof for Incentive Compatibility in Delegation

Proof of Theorem 4.1. All agents act rationally. Hence, at each time step τ, the payment received by any agent $n_i \in \mathcal{N}$ is determined entirely by the first component of the payment rule, since any deviation that triggers an infinite penalty is strictly dominated. Consequently, agents have no incentive to misreport in ways that violate feasibility.

Fix a task $t_i \in \mathcal{T}$, and consider an agent n_i that lies on the critical path for this task. Conditioned on the reported competence vector $(c'_{i,1}, \ldots, c'_{i,k})$, agent n_i remains on the winning path regardless of its delegation decision $v_i^\tau \in r_i^\tau \cup \{n_i\}$. For agents not on the critical path, the payment remains zero irrespective of their delegation choice. The realized payment to agent n_i depends only on the set of agents assigned to task t_i and their reported competences.

Formally, for any two delegation choices v_i^τ of the remaining agents, we have

$$\mathbb{I}\big\{n_i \in W(v_i^\tau, a_{-i}^\tau)\big\} = \mathbb{I}\big\{n_i \in W(\tilde{v}_i^\tau, a_{-i}^\tau)\big\}\,,$$

where $W(\cdot)$ denotes the winning path induced by the corresponding joint action. Thus, delegation preserves the voting weight accumulated by agent n_i, since votes from upstream agents remain unchanged and agent n_i retains its competence contribution on all tasks.

We now analyze the incentive properties of agent n_i under different reporting behaviors.

Case 1: Truthful reporting. Suppose agent n_i reports its true competence on task t_i, i.e., $c'_{i,t_i} = c_{i,t_i}$. Then, for any feasible delegation decision v_i^τ, the utility of agent i satisfies $u_i(a_i^\tau, a_{-i}^\tau) = f(c_{i,t_i}) - f(c_{i-1,t_i})$, which is independent of the choice of v_i^τ. Hence, truthful reporting yields identical utility across all feasible delegation actions.

Case 2: Over-reporting. Suppose agent i inflates its reported competence from its true value c_{i,t_i} to some $c'_{i,t_i} > c_{i,t_i}$, based on its private social information h_i^τ. Such over-reporting is feasible only if there exists a neighbor $n_j \in r_i^\tau$ such that $c^{\tau-1}_{j\to i,t_k} > c_{i,t_i}$. If agent n_i chooses self-delegation, $v_i^\tau = n_i$, then no such informational advantage can be exploited, and thus $c'_{i,t_i} = c_{i,t_i}$. In this case, the utility of agent n_i is $u_i^{\text{self}} = f(c_{i,t_i}) - f(c_{i-1,t_i})$. Now suppose agent n_i delegates to a neighbor $j \in r_i^\tau$ such that $c^{\tau-1}_{j\to i,t_k} > c_{i,t_i}$. Then agent n_i may inflate its reported competence up to $c'_{i,t_i} \le c^{\tau-1}_{j\to i,,t_i}$. In this case, the highest reported competence on task t_i becomes $c^{\tau-1}_{j\to i,,t_i}$, yielding utility $u_i^{\text{del}} = f(c'_{i,t_i}) - f(c_{i-1,t_i})$.

If $\max_{j\in r_i^\tau \cup \{i\}} c^{\tau-1}_{j\to i,t_k} = c_{i,t_i}$, then no such neighbor exists and $c'_{i,t_i} = c_{i,t_i}$, implying $u_i^{\text{self}} = u_i^{\text{del}}$. Otherwise, if there exists a neighbor n_j satisfying $c^{\tau-1}_{j\to i,t_k}$, then delegating to n_j allows agent n_i to increase its reported competence, yielding $u_i^{\text{del}} \ge u_i^{\text{self}}$. Therefore, delegation to a neighbor with higher diffused competence weakly dominates self-voting under over-reporting.

Case 3: Under-reporting. Suppose agent n_i reports $c'_{i,t_i} < c_{i,t_i}$. Then its utility becomes $u_i^{\text{self}} = f(c'_{i,t_i}) - f(c_{i-1,t_i})$, which is strictly smaller than the truthful utility $f(c_{i,t_i}) - f(c_{i-1,t_i})$, since $f(\cdot)$ is strictly increasing. Therefore, under-reporting strictly reduces utility and is never optimal for a rational agent.

Combining all three cases, we conclude that delegating to a neighbor n_j such that $c^{\tau-1}_{j\to i,t_k} > c_{i,t_i}$, constitutes a weakly dominant strategy relative to retaining the vote. This completes the proof. □

A.2 Proof for Continuous Critical Path

Proof of Theorem 4.2. Fix a task $t_k \in \mathcal{T}$. For each agent n_i, let v_i denote its delegated vote, and let $\mathrm{guru}(i)$ be the terminal agent reached by following delegation links from n_i.

Let C_{t_k} be the set of agents critical for t_k, where n_i is critical if, when n_i deviates by voting for itself, the resulting winning path contains (and thus terminates at) n_i.

First, let $n_a, n_b \in C_{t_k}$. Suppose for contradiction that

$$\mathrm{guru}(a) \neq \mathrm{guru}(b).$$

If n_a votes for itself, criticality implies the winning path terminates at n_a. Similarly, if n_b votes for itself, it must terminate at n_b. Since all other agents' actions remain unchanged in each deviation, both outcomes cannot simultaneously correspond to strictly maximal total weight. This contradicts the assumption that both n_a and n_b are critical. Hence,

$$\mathrm{guru}(a) = \mathrm{guru}(b).$$

It remains to prove that C_{t_k} is contiguous along the winning delegation path. Suppose $n_a, n_b \in C_{t_k}$ share the same guru, but neither lies on the delegation path of the other. Consider the deviation where n_a votes for itself. By criticality, the winning path must terminate at n_a, while n_b's contribution remains unchanged. The symmetric argument holds when n_b votes for itself.

Thus, the two deviations produce winning paths ending at different agents with identical contributions from all non-deviators, so both cannot yield strictly maximal weight. This contradiction implies that if two agents in C_{t_k} share the same guru, one must lie on the delegation path of the other. Therefore, C_{t_k} forms a continuous segment of the winning delegation path. □

A.3 Proof of Nash Equilibrium Theorem 4.3

A.3.1 Nash Equilibrium Delegation Rule

Add what the equilibrium rule is here

Fix a Nash equilibrium action profile $(a_i)_{n_i \in \mathcal{N}}$ and let $n_i \in \mathcal{N}$ be an arbitrary agent. After the joint action $a = (a_1, \ldots, a_N)$ is taken and the state transitions to s', agent n_i receives a private observation

$$o_i \;\in\; \mathcal{O}_i,$$

comprising its realized payment together with private social information $\{c^{\tau-1}_{j\to i,\,t_k}\}$, formally:

$$o_i \;=\; \Big\{ \big(j,\; (c_{j\to i,1}, \ldots, c_{j\to i,k}),\; h^i_j\big) \;\Big|\; j \in r_i \cup \{i\} \Big\}.$$

In particular, for each neighbor $n_j \in r_i$, agent n_i observes the diffused competence $c_{j\to i,\,t_i}$ on its assigned task t_i, and observes its own true competence $c^{i\to i}_{t_i} = c_i$. Define

$$\tilde{c}_i \;=\; \max_{j \in r_i} \; c_{j\to i,\,t_i}$$

as the highest diffused competence value observed by n_i across its neighborhood r_i. We establish the equilibrium delegation rule by exhaustive case analysis.

Case 1: $c_i \geq \tilde{c}_i$.

Suppose, toward contradiction, that at equilibrium agent n_i delegates to some neighbor, i.e. $v_i = n_j$ for some $j \in r_i$. By definition of $\tilde{c}_i$, the diffused competence of any such neighbor satisfies

$$c_{j\to i,\,t_i} \;\leq\; \tilde{c}_i \;\leq\; c_i.$$

Hence the guru induced by the delegation path $v_i = n_j$ has reported competence on task t_i that is weakly less than c_i. By the payment rule, agent n_i's reported competence weakly exceeds that of its guru, so its marginal contribution is non-positive and its payment is weakly negative.

In contrast, if agent n_i self-delegates ($v_i = n_i$), the reporting rule yields

$$c^{i\to i}_{t_i} \;=\; c_i,$$

so n_i incurs no negative payment. Since self-delegation weakly dominates delegating to any neighbor, the unique best response is

$$v_i \;=\; n_i.$$

Case 2: $c_i < \tilde{c}_i$**.**

Let

$$n_{j^*} \;\in\; \arg\max_{j \in r_i} \; c_{j \to i,\, t_i}$$

denote a neighbor achieving the maximal diffused competence observed by n_i. We rule out the two suboptimal alternatives.

Subcase 2a: Delegating to a non-maximizing neighbor. If $v_i = n_j$ for some $j \neq j^*$, the utility received by n_i is

$$f(c_{j \to i,\, t_i}) - f\big(c'_{i-1,\, t_i}\big) \;<\; f(c_{j^* \to i,\, t_i}) - f\big(c'_{i-1,\, t_i}\big)\,,$$

since $c_{j \to i,\, t_i} < c_{j^* \to i,\, t_i}$ and f is strictly increasing. Hence delegating to any n_j with $j \neq j^*$ is strictly dominated by delegating to n_{j^*}.

Subcase 2b: Self-delegation. If $v_i = n_i$, the reporting rule yields $c_{t_i}^{i \to i} = c_i$. Since $c_i < \tilde{c}_i = c_{j^* \to i,\, t_i}$, self-delegation produces strictly lower utility than delegating to n_{j^*}.

Combining both subcases, the unique best response is

$$v_i \;=\; j^* \;\in\; \arg\max_{j \in r_i} \; c_{j \to i,\, t_i}.$$

Conclusion.

Since Cases 1 and 2 are exhaustive and mutually exclusive, at any Nash equilibrium action profile $(a_i)_{n_i \in \mathcal{N}}$ every agent $n_i \in \mathcal{N}$ adopts the delegation rule

$$v_i \;=\; \begin{cases} n_i, & \text{if } c_i \geq \tilde{c}_i, \\ \arg\max\limits_{n_j \in r_i} \; c_{j \to i,\, t_i}, & \text{if } c_i < \tilde{c}_i. \end{cases}$$

■

Proof. Fix a Nash equilibrium action profile $(a_i)_{n_i \in \mathcal{N}}$, and fix agent $n_i \in \mathcal{N}$.

Case 1: Diffusion to the critical agents

If agent n_i's private social action h_i fails to diffuse a competence value high enough to the critical agent, then that agent does not choose $v_j = n_i$. Consequently, agent n_i does not lie on the winning path and receives zero payment:

$$u_i(a_i, a_{-i}) = 0.$$

Thus, any equilibrium diffusion must transmit sufficient competence information to the critical agent to secure delegation. The minimum such diffusion is $c_{t_i}^{(2)}$. Applying this inductively across nodes we get the equilibrium reporting strategy as

$$c_{t_i}^{h} = \begin{cases} c_{t_i}^{i}, & v_i = n_i \\ \hat{c}_i, & v_i \neq n_i \end{cases}$$

where $\hat{c}_i = \max_{v \in S(i, v_i) \setminus \arg\max_{u \in S(i, v_i)} c_u} c_v$, $\tilde{d}_i = \max_{j \in r_i} c_{t_i}^{j \to i}$ and $S(i,j) = \{n_k \in \mathcal{N} \mid n_j \in \text{reach}(n_k) \text{ and } n_k \in P,\ \forall P \in \mathcal{P}_{j \to i}\}$, where $\text{reach}(n_i) = \{n_j \in \mathcal{N} \mid \exists \text{ a directed path from } n_i \text{ to } n_j\}$ denotes the set of agents reachable from n_i, and $\mathcal{P}_{j \to i}$ denotes the set of all directed paths from n_j to n_i in the diffusion graph. Intuitively, this means that the most competent reachable agent n_j to agent n_i diffuses just enough information, which is equivalent to most competent reachable node not in the path of n_i, n_j to get the vote of n_i.

Case 2: Diffusion to non-critical agents

Altering diffusion to agents not on the critical path does not change: the identity of the guru, the reported values $c_{t_i}^{i\to i}$ and $c_{t_i}^{(2)}$, or the winning path. Hence,

$$u_i(a_i, a_{-i}) = f(c_{t_i}^{i\to i}) - f(c_{t_i}^{(2)})$$

remains unchanged.

Step 3: Optimality of the diffusion rule.

Any best response diffusion must:

- ensure delegation by the critical agent;
- not reduce the agent's own payment.

The diffusion rule

$$c_{i\to j,t_j} = \begin{cases} c_{i\to i,t_i}, & \text{if } c_{j\to i,t_j} > c_{i\to i,t_j}, \\ c_{j\to i,t_j}, & \text{if } c_{j\to i,t_j} \le c_{i\to i,t_j}. \end{cases}$$

satisfies both conditions. It guarantees delegation from all agents with higher competence while preserving the payment term $f(c_{t_i}^{i\to i}) - f(c_{t_i}^{(2)})$. Hence, this diffusion strategy is optimal at Nash equilibrium

As in the equilibrium described if agent n_i delegates its vote to g_i, then every agent reachable from n_i whose task coincides with that of n_i also delegates to g_i. Consequently, all such agents lie on the same feasible path, if any at equilibrium. This ensures that agent n_i can report the highest achievable competence on all auxiliary tasks, since all reachable agents contributing to those tasks are contained within a single path. Any deviation from this delegation structure strictly reduces the set of agents on agent n_i's feasible path. As a result, the maximal reported competence attainable on auxiliary tasks weakly decreases. Therefore, no such deviation can improve agent n_i's outcome, and the preceding analysis remains unaffected.

□

A.4 Best Response for Single-Task Request

Lemma A.1 (Best Response for Single-Task Request)**.** ***Diffusion and Delegation.*** *Let agent $n_i \in \mathcal{N}$ have primary task t_i, neighbors r_i, and true competence c_{i,t_i} on task t_i. At round $\tau - 1$, agent n_i observes from its private observation, for each $n_j \in r_i$, the diffused competence value*

$$c_{j\to i,t_i}^{\tau-1}, \text{ with } c_{i\to i,t_i}^{\tau-1} := c_{i,t_i}.$$

We define

$$c^* = \max_{n_j \in r_i \cup \{n_i\}} c_{j\to i,t_i}^{\tau-1}, n_{j^*} \in \arg\max_{n_j \in r_i \cup \{n_i\}} c_{j\to i,t_i}^{\tau-1}.$$

Then there exists a best response of agent n_i at round τ such that:

(Diffusion) *The private social action $h_i^\tau \in \mathcal{H}_i$ satisfies, for each $n_j \in r_i$,*

$$c_{i\to j,,t_i}^{\tau} = \begin{cases} \varnothing, & n_j = n_{j^*}, \\ \min\left(c_{j\to i,t_i}^{\tau-1} + \delta,\ c^*\right), & n_j \in r_i \setminus \{j^*\} \text{ and } c_{j\to i,t_i}^{\tau-1} < c^*, \\ \varnothing, & \text{otherwise}, \end{cases}$$

where $\delta > 0$ is a fixed diffusion increment, and $c_{i\to j,t_i}^{\tau}$ denotes the competence component on task t_i transmitted by agent n_i to agent n_j through h_i^τ.

(Delegation) *The delegation component of agent n_i's action satisfies*

$$v_i^\tau \in \arg\max_{v \in r_i \cup \{i\}} c_{v\to i,,t_i}^{\tau-1}.$$

In particular, delegating to an agent attaining the maximal observed diffused competence value (or to itself, if it attains the maximum), together with the diffusion rule above, constitutes a best response for agent n_i at round τ.

Delegation Best Response Fix any agent $n_i \in \mathcal{N}$, and let r_i denote its neighbors. From its private observation, agent n_i observes, for each $n_j \in r_i$, the diffused competence value $c_{j\to i,t_i}$ on its own task t_i, and it observes its own true competence $c_{i\to i,t_i} = c_i$. We define $\hat{c}_i = \max\Big(c_i,\ \max_{j\in r_i} c_{j\to i,t_i}\Big)$, the highest competence value available to agent n_i. Let agent n_i choose its delegation component $v_i^* \in \arg\max_{v\in r_i\cup\{i\}} c_{t_i}^{v\to i}$. We show that this delegation rule is optimal: for any alternative delegation $v_i' \neq v_i^*$, i.e. $u_i\big(a_i(v_i^*), a_{-i}\big) \geq u_i\big(a_i(v_i'), a_{-i}\big)$, where $a_i(v)$ denotes agent n_i's action with delegation component v and all other components fixed.

Agent n_i receives a strictly positive payment if and only if it lies on the critical path of the winning delegation tree.

Case 1: Agent n_i is not critical under v_i^*.

Any alternative delegation $v_i' \neq v_i^*$ cannot move agent n_i onto the winning path, and hence

$$u_i\big(a_i(v_i'), a_{-i}\big) \leq 0 = u_i\big(a_i(v_i^*), a_{-i}\big).$$

Case 2: Agent n_i is critical under v_i^*.

Then agent n_i lies on the winning path, and its realized payment equals its marginal contribution:

$$u_i\big(a_i(v_i^*), a_{-i}\big) = f(\hat{c}_i) - f(\hat{c}_i^{(2)}),$$

where $\hat{c}_i^{(2)}$ denotes the competence value of the agent immediately preceding n_i on the critical path, if any. Consider any alternative delegation $v_i' \neq v_i^*$. By definition of v_i^*, $c_{v_i'\to i,t_i} \leq c_{v_i^*\to i,t_i} = \hat{c}_i$. Since agent n_i remains critical under v_i', its payment is

$$u_i\big(a_i(v_i'), a_{-i}\big) = f(c_{v_i'\to i,t_i}) - f(\hat{c}_i^{(2)}).$$

Because f is monotone non-decreasing, $f(c_{v_i'\to i,t_i}) \leq f(\hat{c}_i)$. Hence,

$$u_i\big(a_i(v_i'), a_{-i}\big) \leq f(\hat{c}_i) - f(\hat{c}_i^{(2)}) = u_i\big(a_i(v_i^*), a_{-i}\big).$$

Therefore, any deviation from v_i^* weakly decreases agent n_i's payment, with strict inequality whenever $c_{v_i'\to i,t_i} < c_{v_i^*\to i,t_i}$.

Thus, delegating to $v_i^* \in \arg\max_{v\in r_i\cup\{i\}} c_{v\to i,t_i}$ is optimal for agent n_i. □

Diffusion Best Response From its private observation at round $\tau - 1$, agent n_i observes

$$\{\, c_{j\to i,t_i}^{\tau-1} \mid n_j \in r_i \,\},\ \text{with } c_{i\to i,t_i}^{\tau-1} := c_{i,t_i}.$$

We define

$$c^* = \max_{j\in r_i\cup\{i\}} c_{j\to i,t_i}^{\tau-1}, n_{j^*} \in \arg\max_{j\in r_i\cup\{i\}} c_{j\to i,t_i}^{\tau-1}.$$

Agent n_i chooses its private social action $h_i^\tau \in \mathcal{H}_i$, which determines the diffused values $c_{t_i}^{i\to j,\,\tau}$ for each $n_j \in r_i$. We analyze agent n_i's diffusion incentives neighbor by neighbor.

Case 1: No diffusion to n_{j^*}.

By definition of n_{j^*},

$$c_{j^*\to i,t_i}^{\tau-1} = c^*.$$

Hence agent n_{j^*} already observes a competence value weakly higher than any value agent n_i can diffuse. Therefore, under any feasible diffusion choice, agent n_{j^*} does not update its delegation decision in response to h_i^τ. Thus, in any best response,

$$c_{i\to j^*,t_i}^{\tau} = \varnothing.$$

Case 2: Diffusion to non-maximal neighbors.

We fix any $n_j \in r_i \setminus \{n_{j^*}\}$,

Case 2.1: $c^{\tau-1}_{j\to i,t_i} \geq c^*$.

This case is impossible by definition of c^*.

Case 2.2: $c^{\tau-1}_{j\to i,t_i} < c^*$.

If agent j is non-critical, changing $c^{\tau}_{i\to j,t_i}$ does not affect agent n_i's payment. If agent n_j is critical, failing to induce delegation from n_j implies that agent n_i does not lie on the winning path, yielding zero payment. Hence, whenever n_j is critical, agent n_i must ensure $c^{\tau}_{i\to j,t_i} \geq c^{\tau-1}_{j\to i,t_i}$ to secure n_j's delegation. However, diffusing a value strictly larger than c^* cannot increase agent n_i's payment, since c^* is the maximal competence value available in n_i's neighborhood. Therefore, any diffusion above c^* is weakly dominated. Thus, an optimal diffusion choice satisfies

$$c^{\tau}_{i\to j,t_i} = \min\big(c^{\tau-1}_{j\to i,t_i} + \delta,\ c^*\big)\,,$$

where $\delta > 0$ is the minimal diffusion increment.

Therefore, the diffusion policy

$$c^{i\to j,\,\tau}_{t_i} = \begin{cases} \varnothing, & n_j = n_{j^*}, \\ \min\big(c^{\tau-1}_{j\to i,t_i} + \delta,\ c^*\big)\,, & n_j \in r_i \setminus \{n_{j^*}\} \text{ and } c^{\tau-1}_{j\to i,t_i} < c^*, \\ \varnothing, & \text{otherwise}, \end{cases}$$

constitutes a best response for agent n_i at round τ. □

B `AgentSociety` Pseudocode

Algorithm 1: AgentSociety: Mechanism for consensus based request routing among self-interested heterogeneous agents

Input : Multi-task query Q, intrinsic competence C_i, initial diffusion state $\mathcal{S}_i^{\text{diff}} = \{\hat{C}_{k\to i}\}_{k\in\mathcal{N}(i)}$

Output : Optimal path $\mathcal{P}^*$, Updated diffusion state $\mathcal{S}_i^{\text{diff}'}$

`// Phase 1: Delegation and strategic competence reporting`

foreach *agent* $i \in \mathcal{T}_{t_k}$ ***in parallel*** **do**

  Observe current diffusion state $\mathcal{S}_i^{\text{diff}}$ (signals received from neighbors);

  Submit strategic action $a_i = (\mathbf{c}_i', v_i, r_i)$ to the ledger where:

  $\mathbf{c}_i' = \max(C_i, \max_k \hat{C}_{k\to i})$; `// Signal derived from internal competence and received diffusion`

  $v_i = \arg\max_{k\in\mathcal{N}(i)} \hat{C}_{k\to i}$; `// Delegate to the most competent signaling neighbor`

`// Phase 2: Consensus based request routing path`

foreach *task* $t_j \in Q$ **do**

  Identify gurus $\mathcal{G}_j = \{g \in \mathcal{T}_{t_j} \mid v_g = g\}$;

  Compute transitive vote counts $V(g) = \sum_{i:v_i\to g} 1$ via delegation graph traversal;

Initialize feasible paths $\mathbb{F} = \emptyset$;

foreach $g \in \mathcal{G}_1$ **do**

  Select bridge $d^* = \arg\max_{d\in\mathcal{D}(g)} \sum_{j>1} c_d^{t_j}$;

  if *bridge* d^* *connects to guru* $g' \in \mathcal{G}_{k+1}$ **then**

    $\mathbb{F} \leftarrow \mathbb{F} \cup \{\mathcal{P}_{g\to g'}\}$;

$\mathcal{P}^* = \arg\max_{\mathcal{P}\in\mathbb{F}} \sum_{g\in\mathcal{P}} V(g)$;

`// Phase 3: Payoff Calculation`

foreach *agent* $v \in \mathcal{P}^*$ **do**

  if v *is critical ;* `// via counterfactual audit`

  then

    $P_v = p^{(t_j)}(v) - p^{\text{inf}}(v) - p^{\text{mis}}(v)$;

`// Phase 4: Social Diffusion and State Update`

foreach *agent* $i \in \mathcal{N}$ ***in parallel*** **do**

  Observe realized reward P_i and transition state s_{t+1};

  if $P_i <$ *Threshold (Sub-optimal Outcome)* **then**

    ; `// Modify diffusion action to garner future influence`

    $\hat{C}_{i\to j} \leftarrow C_i + \epsilon$ for $j \in \mathcal{N}(i)$;

  else

    $\hat{C}_{i\to j} \leftarrow C_i$; `// Maintain current signaling level`

  Update Diffusion State $\mathcal{S}_i^{\text{diff}'} = \{\hat{C}_{k\to i}\}_{k\in\mathcal{N}(i)}$ based on new signals from neighbors;

  Update history $\Pi_i \leftarrow \Pi_i \cup \{P_i\}$;

C Configurations and additional experimental results

C.1 Configuration for diffusion payment analysis Fig. 2 (right).

```
CUSTOM_GRAPH_CONFIG = {
    'nodes': [
        {'id': 1, 'primary_task': 1, 'intrinsic_competence': {1: 0.7, 2:
            0, 3: 0}},
        {'id': 2, 'primary_task': 1, 'intrinsic_competence': {1: 0.7, 2:
            0, 3: 0}},
        {'id': 3, 'primary_task': 1, 'intrinsic_competence': {1: 0.6, 2:
            0, 3: 0}},
        {'id': 4, 'primary_task': 1, 'intrinsic_competence': {1: 0, 2:
            0.4, 3: 0}},
        {'id': 5, 'primary_task': 1, 'intrinsic_competence': {1: 0, 2:
            0.5, 3: 0}},
        {'id': 6, 'primary_task': 1, 'intrinsic_competence': {1: 0, 2:
            0.4, 3: 0}},
        {'id': 7, 'primary_task': 1, 'intrinsic_competence': {1: 0, 2:
            0.6, 3: 0}},
        {'id': 8, 'primary_task': 1, 'intrinsic_competence': {1: 0, 2:
            0.6, 3: 0}},
        {'id': 9, 'primary_task': 1, 'intrinsic_competence': {1: 0, 2:
            0.5, 3: 0}},
        {'id': 10, 'primary_task': 2, 'intrinsic_competence': {1: 0, 2:
            0.9, 3: 0}},
        {'id': 11, 'primary_task': 2, 'intrinsic_competence': {1: 0, 2:
            0.7, 3: 0}},
        {'id': 12, 'primary_task': 2, 'intrinsic_competence': {1: 0, 2:
            0.6, 3: 0}},
        {'id': 13, 'primary_task': 2, 'intrinsic_competence': {1: 0, 2:
            0.6, 3: 0}},
        {'id': 14, 'primary_task': 3, 'intrinsic_competence': {1: 0, 2: 0,
            3: 0.5}}
    ],
    'edges': [
        (1, 4), (2, 6), (3, 7), (4, 8), (5, 6), (6, 11), (8, 9),
        (9, 10), (10, 11), (11, 12), (11, 13), (8, 14), (11, 14), (7, 14)
    ]
}
```

C.2 Configurations for LLM agent social intelligence characterization

In all LLM agent experiments, Node 2 is assigned as the LLM agent.

C.2.1 Configuration 1

The dynamics for this configuration are presented in the main text in Fig. 3.

```
CUSTOM_GRAPH_CONFIG_1 = {
    'nodes': [
        {'id': 1, 'intrinsic_competence': {1: 0.6}},
        {'id': 2, 'intrinsic_competence': {1: 0.3}},
        {'id': 3, 'intrinsic_competence': {1: 0.5}}
    ],
    'edges': [
        (1, 2),
        (2, 3),
    ]
}
```

We consider additional configurations to show that our mechanism is configuration-agnostic.

C.2.2 Configuration 2 and its corresponding dynamics

```
CUSTOM_GRAPH_CONFIG_2 = {
    'nodes': [
        {'id': 1, 'intrinsic_competence': {1: 0.6}},
        {'id': 2, 'intrinsic_competence': {1: 0.4}},
        {'id': 3, 'intrinsic_competence': {1: 0.5}},
        {'id': 4, 'intrinsic_competence': {1: 0.7}},
        {'id': 5, 'intrinsic_competence': {1: 0.3}},
        {'id': 6, 'intrinsic_competence': {1: 0.3}},
    ],
    'edges': [
        (1, 2),
        (2, 3),
        (3, 4),
        (2, 5),
        (2, 6)
    ]
}
```

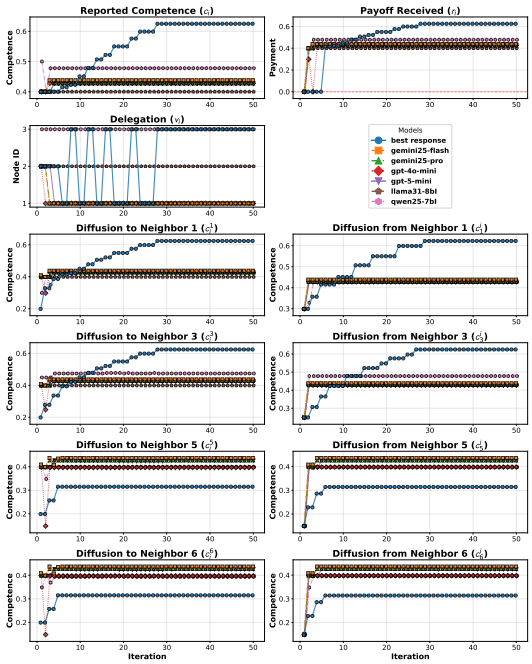


Figure 4: LLM Agent dynamics in `AgentSociety` benchmarked against best response for configuration 2

C.2.3 Configuration 3 and its corresponding dynamics

```
CUSTOM_GRAPH_CONFIG_3 = {
    'nodes': [
        {'id': 1, 'intrinsic_competence': {1: 0.7}},
        {'id': 4, 'intrinsic_competence': {1: 0.5}},
        {'id': 3, 'intrinsic_competence': {1: 0.4}},
        {'id': 2, 'intrinsic_competence': {1: 0.65}},
        {'id': 5, 'intrinsic_competence': {1: 0.4}},
        {'id': 6, 'intrinsic_competence': {1: 0.6}},
        {'id': 7, 'intrinsic_competence': {1: 0.5}},


    ],
    'edges': [
        (1, 4),
```

```
        (4, 3),
        (3, 2),
        (2, 5),
        (5, 6),
        (6, 7)
    ]
}
```

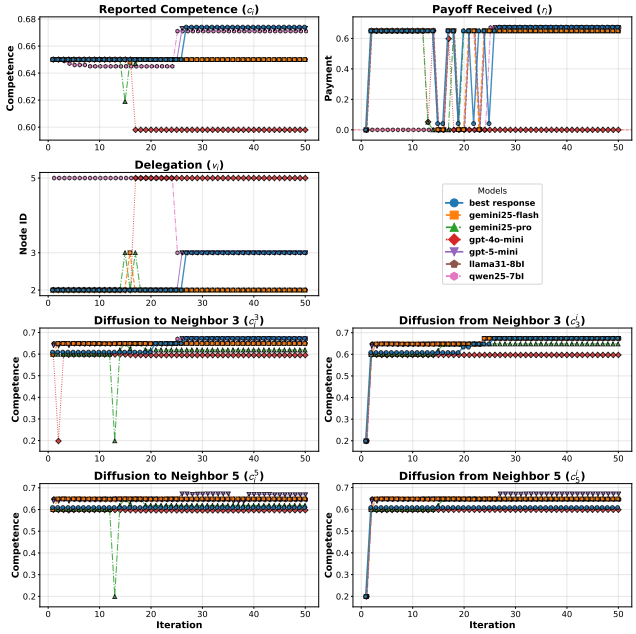


Figure 5: LLM Agent dynamics in `AgentSociety` benchmarked against best response for configuration 3

D Experimental Details

We provide details on each of our experimental result here. All experiments were run on CPU with 16 GB RAM with LLM inferences accessed through API calls. Delegation and information diffusion decisions involve two additional calls to each LLM agent per task. In total, num-agents*2 additional calls per task represent the token overhead and cost, while the latency of the task is dictated by the maximum among the agents in the graph. As we leverage multiple heterogeneous LLMs, the latency and token costs vary by provider. Our average token overhead for delegation is 1200 input tokens and 20 output tokens while for diffusion the token overhead is 1400 input tokens and 45 output tokens, with costs for both in range of USD 0.0001 - USD 0.001 per call. Our highest latency is Gemini's API driving about 14s per call.

D.1 Mechanism Characterization

We elaborate on the experiment details to generate Fig. 2 (left). We characterize the mechanism on a population of 17 nodes, where the topology is sampled randomly by partitioning agents into two tasks. Each agent adopts the best response strategy discussed in § A.4. Each agent's intrinsic competence c is modeled using a clipped, squashed log-normal distribution ($z \sim \text{LogNormal}(-1.0, 1.0)$, $c = \text{clip}(z/(1+z), 0, 1)$) to reflect heavy-tailed expertise inequality, where a minority of agents possess high competence. The simulation executes for $T = 50$ iterations. Each iteration follows a six-stage pipeline: (1) Delegation via the best response strategy; (2) Path Computation for root-to-guru chains; (3) Winner Determination based on aggregate votes; (4) Critical-node Analysis to identify essential agents; (5) Information Diffusion, where non-critical nodes diffuse competence values (6) Competence Update for future competence reports increment by $\Delta_r = 0.02$ subject to the

neighborhood's diffused upper bound. To ensure statistical robustness, we perform 100 independent replications across different random graph seeds and report the mean and standard deviation for five key metrics. We track the Strategic Competence Report ($\bar{c}_t^{\text{rep}}$) to measure population-level reporting, User Realized Competence ($\bar{c}_t^{\text{real}}$) for the quality of service delivered by winning gurus, and Max Competence as the theoretical upper bound. Additionally, we log the Intrinsic Competence and the Payoff to Intermediaries ($\bar{p}_t$) to quantify the cost of delegation distributed to non-guru agents along the winning paths, and plot these in Fig. 2 (left).

D.2 LLM Agent Social Intelligence in `AgentSociety`

We elaborate on the experiment details to generate Fig. 3 (left). We evaluate each LLM agent's social intelligence by introducing the LLM into `AgentSociety` with other nodes in the topography operating on the best response strategy. At each time step, the LLM agent must decide its delegation and diffusion behavior using natural language reasoning given the system context of the `AgentSociety` as well as prior history. Such an environment isolates the behavioral effects of replacing a rigid algorithmic participant with a free-form reasoner. We utilize a suite of static graph configurations (3–7 nodes) covering diverse topologies. At each iteration, the LLM agent is prompted twice—once for delegation and once for diffusion using their respective prompts, please see § E for the prompts.

We benchmark each of the 6 LLMs presented in Fig. 3 (left) with temperature 0.0 (except gpt-5-mini that only takes 1.0) and a 4096 token limit. To ensure longitudinal coherence, the agent's full per-iteration memory (previous delegation targets, payments received, and diffusion records) is fed back into the subsequent prompt, allowing the LLM to operate on its own historical context. The LLM agent observes payoffs given by Eq. 1 with α set to 100. Since all other nodes remain deterministic, any deviation from best response provides a clean ablation of the LLM's social intelligence impact on mechanism. We quantify LLM behavioral divergence from the best-response equilibrium across two granularities in the presence of other best response agents: decision-level divergence, representing the cumulative difference in actions given a fixed state, and trajectory-level divergence, representing the cumulative shift in the evolved state itself. These deviations are measured over multiple configurations through the percentage overlap in delegation choices and the mean absolute error (MAE) of diffused information and reported competence in Fig. 3 (left).

D.3 Collaborative Performance

In each of these experiments, we consider 6 models instantiated in `AgentSociety` with a fixed (randomly generated) graph topology (for uniformity). The capability vectors are of the dimension as domains present in the datasets and estimated using a 30% train split and the evaluation is done on the remaining 70%.

MMLU-Pro We utilize the TIGER-Lab/MMLU-Pro dataset, generating 5-shot predictions for each model via the official answer-extraction protocol and mapped to their 14 subdomains. The evaluation panel includes six diverse models: Llama-3.1-8B-Instruct, Qwen1.5-7B-Chat, Claude-3.5-Sonnet, Gemini-1.5-Pro-002, Gemini-2.0-Flash-Exp, and GPT-4o-Mini.

```
Task IDs: 1=biology, 2=business, 3=chemistry, 4=computer science,
    5=economics, 6=engineering, 7=health, 8=history,
9=law, 10=math, 11=other, 12=philosophy, 13=physics, 14=psychology

MMLU_PRO_CONFIG = {
    'nodes': [
        {'id': 1, 'intrinsic_competence': {1: 0.8951, 2: 0.7810, 3:
            0.8208, 4: 0.8110, 5: 0.8160, 6: 0.6253, 7: 0.7676,
8: 0.7434, 9: 0.5659, 10: 0.8574, 11: 0.7453, 12: 0.6884, 13: 0.8073,
    14: 0.7806}},  # gemini-2.0-flash-exp
        {'id': 2, 'intrinsic_competence': {1: 0.8357, 2: 0.7111, 3:
            0.6173, 4: 0.6951, 5: 0.7033, 6: 0.4109, 7: 0.7064,
8: 0.5855, 9: 0.3682, 10: 0.7259, 11: 0.6775, 12: 0.5678, 13: 0.6262,
    14: 0.7210}, },  # gpt-4o-mini
        {'id': 3, 'intrinsic_competence': {1: 0.8881, 2: 0.8032, 3:
            0.6350, 4: 0.6951, 5: 0.7982, 6: 0.5943, 7: 0.7706,
8: 0.7500, 9: 0.5591, 10: 0.5444, 11: 0.7127, 12: 0.7186, 13: 0.8189,
    14: 0.8276}, },  # gemini-1.5-pro-002
```

```
        {'id': 4, 'intrinsic_competence': {1: 0.9196, 2: 0.7905, 3:
            0.8031, 4: 0.8110, 5: 0.8309, 6: 0.6408, 7: 0.7829,
  8: 0.7697, 9: 0.6568, 10: 0.7537, 11: 0.7724, 12: 0.7236, 13: 0.7592,
      14: 0.8056}, },  # claude-3.5-sonnet
        {'id': 5, 'intrinsic_competence': {1: 0.6783, 2: 0.4698, 3:
            0.3761, 4: 0.4756, 5: 0.5341, 6: 0.2946, 7: 0.5352,
  8: 0.4605, 9: 0.2500, 10: 0.4593, 11: 0.4743, 12: 0.4874, 13: 0.3815,
      14: 0.5893}, },  #  Meta-Llama-3_1-8B-Instruct
        {'id': 6, 'intrinsic_competence': {1: 0.5000, 2: 0.2540, 3:
            0.1261, 4: 0.2744, 5: 0.4125, 6: 0.1628, 7: 0.2355,
  8: 0.3158, 9: 0.1773, 10: 0.2796, 11: 0.3388, 12: 0.2362, 13: 0.1599,
      14: 0.4295}, },  # Qwen1.5-7B-Chat
      ],
      'edges': [(1, 2), (2, 3), (3, 4), (2, 5), (2, 6)],
  }
```

Open LLM Leaderboard v2 To prevent data duplication with MMLU-Pro above, we restrict this panel to the 15 non-MMLU sub-tasks, including IFEval, GPQA-Main, MATH (Algebra and Intermediate Algebra Hard), MuSR, and ten BBH sub-tasks. The model suite for this benchmark spans three capacity tiers: Llama-3.1-70B-Instruct, Tulu-3-70B, phi-4, Llama-3.1-8B-Instruct, Yi-1.5-34B-Chat, and Qwen2.5-Coder-7B-Inst.

```
Task IDs: 1=ifeval, 2=gpqa_main, 3=math_algebra_hard,
    4=math_intermediate_algebra_hard, 5=musr_object_placements,
  6=bbh_boolean_expressions, 7=bbh_date_understanding,
      8=bbh_disambiguation_qa, 9=bbh_formal_fallacies,
  10=bbh_geometric_shapes, 11=bbh_hyperbaton,
      12=bbh_logical_deduction_five_objects,
  13=bbh_logical_deduction_seven_objects,
      14=bbh_logical_deduction_three_objects, 15=bbh_movie_recommendation

  OPENLLM_V2_CONFIG = {
      'nodes': [
        {'id': 1, 'intrinsic_competence': {1: 0.8642, 2: 0.3209, 3:
            0.0217, 4: 0.0000, 5: 0.2368, 6: 0.8933, 7: 0.5867,
  8: 0.7067, 9: 0.7867, 10: 0.2800, 11: 0.7467, 12: 0.5600, 13: 0.4800,
      14: 0.9067, 15: 0.7733}, 'primary_task': 1},  # Llama-3.1-70B-Instruct
        {'id': 2, 'intrinsic_competence': {1: 0.5679, 2: 0.3433, 3:
            0.4239, 4: 0.0714, 5: 0.3026, 6: 0.8933, 7: 0.5600,
  8: 0.7600, 9: 0.6267, 10: 0.4400, 11: 0.6933, 12: 0.5333, 13: 0.4400,
      14: 0.7867, 15: 0.6667}, 'primary_task': 1},  # Yi-1.5-34B-Chat
        {'id': 3, 'intrinsic_competence': {1: 0.7037, 2: 0.3209, 3:
            0.0000, 4: 0.0000, 5: 0.2105, 6: 0.8933, 7: 0.7200,
  8: 0.6800, 9: 0.7467, 10: 0.5200, 11: 0.8133, 12: 0.6400, 13: 0.5467,
      14: 0.9200, 15: 0.6533}, 'primary_task': 1},  # phi-4
        {'id': 4, 'intrinsic_competence': {1: 0.6111, 2: 0.3134, 3:
            0.0761, 4: 0.0000, 5: 0.2632, 6: 0.9333, 7: 0.3733,
  8: 0.5467, 9: 0.5867, 10: 0.5733, 11: 0.5333, 12: 0.4000, 13: 0.3067,
      14: 0.7333, 15: 0.6267}, 'primary_task': 1},  # Qwen2.5-Coder-7B-Inst
        {'id': 5, 'intrinsic_competence': {1: 0.8457, 2: 0.3806, 3:
            0.0000, 4: 0.0000, 5: 0.3289, 6: 0.8133, 7: 0.5600,
  8: 0.6533, 9: 0.6533, 10: 0.3333, 11: 0.6933, 12: 0.3600, 13: 0.3200,
      14: 0.8267, 15: 0.8267}, 'primary_task': 1},  # Tulu-3-70B
        {'id': 6, 'intrinsic_competence': {1: 0.4691, 2: 0.3433, 3:
            0.2826, 4: 0.0119, 5: 0.3421, 6: 0.7600, 7: 0.3733,
  8: 0.4800, 9: 0.4933, 10: 0.3600, 11: 0.6800, 12: 0.3867, 13: 0.3733,
      14: 0.6400, 15: 0.6267}, 'primary_task': 1},  # Llama-3.1-8B-Instruct
      ],
      'edges': [(1, 2), (2, 3), (3, 4), (2, 5), (2, 6)],
  }
```

SWE-bench . We leverage trajectories from princeton-nlp/SWE-bench-Verified test split. Evaluation domains are defined by the top-5 resolved repositories. We benchmark two distinct 6-agent configurations: a Strong/Generalist panel (comprising SWE-agent-Claude-4, Kimi-K2, Augment-Agent, Claude-3.5-Sonnet, DeepSeek-V3, SWE-agent-GPT-4o) and a Weaker/Complementary panel

(including Enginelabs, Claude-3.5-Haiku, GRU, Codeshellagent-Gemini, SWERL-Llama3-70B, and Amazon-Nova-Premier).

```
Task IDs: 1=django/django, 2=matplotlib/matplotlib,
    3=scikit-learn/scikit-learn, 4=sphinx-doc/sphinx, 5=sympy/sympy

SWEBENCH_STRONG_CONFIG = {
    'nodes': [
        {'id': 1, 'intrinsic_competence': {1: 0.6848, 2: 0.5385, 3:
            0.6667, 4: 0.4118, 5: 0.7000}, },  # Kimi-K2
        {'id': 2, 'intrinsic_competence': {1: 0.5000, 2: 0.3846, 3:
            0.6667, 4: 0.2941, 5: 0.5667}, }, # Claude-3.5-Sonnet
        {'id': 3, 'intrinsic_competence': {1: 0.6413, 2: 0.4615, 3:
            0.7500, 4: 0.4706, 5: 0.6667}, }, # Augment-Agent
        {'id': 4, 'intrinsic_competence': {1: 0.7065, 2: 0.4615, 3:
            0.6667, 4: 0.5882, 5: 0.6000}, }, # SWE-agent-Claude-4
        {'id': 5, 'intrinsic_competence': {1: 0.5000, 2: 0.4615, 3:
            0.3333, 4: 0.3529, 5: 0.3000}, }, # DeepSeek-V3
        {'id': 6, 'intrinsic_competence': {1: 0.2609, 2: 0.0000, 3:
            0.2500, 4: 0.0000, 5: 0.2667}, }, # SWE-agent-GPT-4o
    ],
    'edges': [(1, 2), (2, 3), (3, 4), (2, 5), (2, 6)],
}
```

```
Task IDs: 1=django/django, 2=matplotlib/matplotlib,
    3=scikit-learn/scikit-learn, 4=sphinx-doc/sphinx, 5=sympy/sympy

 SWEBENCH_WEAK_CONFIG = {
    'nodes': [
        {'id': 1, 'intrinsic_competence': {1: 0.5000, 2: 0.5385, 3:
            0.5833, 4: 0.4118, 5: 0.5000}, }, # Codeshellagent-Gemini
        {'id': 2, 'intrinsic_competence': {1: 0.4674, 2: 0.3846, 3:
            0.7500, 4: 0.3529, 5: 0.4333}, }, # GRU
        {'id': 3, 'intrinsic_competence': {1: 0.4674, 2: 0.3077, 3:
            0.8333, 4: 0.1765, 5: 0.5333}, }, # Claude-3.5-Haiku
        {'id': 4, 'intrinsic_competence': {1: 0.6413, 2: 0.3846, 3:
            0.7500, 4: 0.0000, 5: 0.6667}, }, # Enginelabs
        {'id': 5, 'intrinsic_competence': {1: 0.4674, 2: 0.4615, 3:
            0.5833, 4: 0.2941, 5: 0.3667}, }, # Amazon-Nova-Premier
        {'id': 6, 'intrinsic_competence': {1: 0.4239, 2: 0.4615, 3:
            0.7500, 4: 0.0588, 5: 0.3667}, }, # SWERL-Llama3-70B
    ],
    'edges': [(1, 2), (2, 3), (3, 4), (2, 5), (2, 6)],
}
```

D.4 Multi-Task Collaborative Performance

We evaluate multi-task delegation mechanism using evaluation traces from the open-llm-leaderboard/model-details dataset. The model suite spans three capacity tiers—including Large (Llama-3.1-70B-Instruct, Tulu-3-70B, phi-4), Mid (Yi-1.5-34B-Chat), and Small (Llama-3.1-8B-Instruct, Qwen2.5-Coder-7B-Inst) — evaluated across 5 reasoning domains from the IFEval, MATH, and BBH suites. To isolate the routing component's performance, we select these domains where no single model dominates, partitioning each into a 40% train split for empirical competence estimation and a 60% held-out test split. The experimental graph consists of 30 nodes (model-domain pairs) in a randomly generated topology. We benchmark multi-task combinations by running the delegation mechanism for 50 iterations with 1,000 sampled test questions per domain to ensure robust performance metrics. The `AgentSociety` mechanism runs similarly to D.1.

```
MULTI_TASK_CONFIG = {
    'nodes': [
        # ifeval (task=1)
        {'id': 1,  'intrinsic_competence': {1: 0.8611, 2: 0, 3: 0, 4: 0,
            5: 0}, 'primary_task': 1},  #
Llama-3.1-70B-Instruct
```

```
        {'id': 2,  'intrinsic_competence': {1: 0.8472, 2: 0, 3: 0, 4: 0,
            5: 0}, 'primary_task': 1},  # Tulu-3-70B
        {'id': 3,  'intrinsic_competence': {1: 0.6991, 2: 0, 3: 0, 4: 0,
            5: 0}, 'primary_task': 1},  # phi-4
        {'id': 4,  'intrinsic_competence': {1: 0.5741, 2: 0, 3: 0, 4: 0,
            5: 0}, 'primary_task': 1},  # Yi-1.5-34B-Chat
        {'id': 5,  'intrinsic_competence': {1: 0.6111, 2: 0, 3: 0, 4: 0,
            5: 0}, 'primary_task': 1},  # Qwen2.5-Coder-7B-Inst
        {'id': 6,  'intrinsic_competence': {1: 0.4769, 2: 0, 3: 0, 4: 0,
            5: 0}, 'primary_task': 1},  # Llama-3.1-8B-Instruct
        # math_algebra_hard (task=2)
        {'id': 7,  'intrinsic_competence': {1: 0, 2: 0.0000, 3: 0, 4: 0,
            5: 0}, 'primary_task': 2},  # Tulu-3-70B
        {'id': 8,  'intrinsic_competence': {1: 0, 2: 0.4180, 3: 0, 4: 0,
            5: 0}, 'primary_task': 2},  # Yi-1.5-34B-Chat
        {'id': 9,  'intrinsic_competence': {1: 0, 2: 0.2869, 3: 0, 4: 0,
            5: 0}, 'primary_task': 2},  # Llama-3.1-8B-Instruct
        {'id': 10, 'intrinsic_competence': {1: 0, 2: 0.0000, 3: 0, 4: 0,
            5: 0}, 'primary_task': 2},  # phi-4
        {'id': 11, 'intrinsic_competence': {1: 0, 2: 0.0574, 3: 0, 4: 0,
            5: 0}, 'primary_task': 2},  # Qwen2.5-Coder-7B-Inst
        {'id': 12, 'intrinsic_competence': {1: 0, 2: 0.0164, 3: 0, 4: 0,
            5: 0}, 'primary_task': 2},  # Llama-3.1-70B-Instruct
        # bbh_geometric_shapes (task=3)
        {'id': 13, 'intrinsic_competence': {1: 0, 2: 0, 3: 0.5800, 4: 0,
            5: 0}, 'primary_task': 3},  # Qwen2.5-Coder-7B-Inst
        {'id': 14, 'intrinsic_competence': {1: 0, 2: 0, 3: 0.4600, 4: 0,
            5: 0}, 'primary_task': 3},  # Yi-1.5-34B-Chat
        {'id': 15, 'intrinsic_competence': {1: 0, 2: 0, 3: 0.4800, 4: 0,
            5: 0}, 'primary_task': 3},  # phi-4
        {'id': 16, 'intrinsic_competence': {1: 0, 2: 0, 3: 0.3300, 4: 0,
            5: 0}, 'primary_task': 3},  # Tulu-3-70B
        {'id': 17, 'intrinsic_competence': {1: 0, 2: 0, 3: 0.2800, 4: 0,
            5: 0}, 'primary_task': 3},  # Llama-3.1-70B-Instruct
        {'id': 18, 'intrinsic_competence': {1: 0, 2: 0, 3: 0.3500, 4: 0,
            5: 0}, 'primary_task': 3},  # Llama-3.1-8B-Instruct
        # bbh_disambiguation_qa (task=4)
        {'id': 19, 'intrinsic_competence': {1: 0, 2: 0, 3: 0, 4: 0.7900,
            5: 0}, 'primary_task': 4},  # Yi-1.5-34B-Chat
        {'id': 20, 'intrinsic_competence': {1: 0, 2: 0, 3: 0, 4: 0.7000,
            5: 0}, 'primary_task': 4},  # Llama-3.1-70B-Instruct
        {'id': 21, 'intrinsic_competence': {1: 0, 2: 0, 3: 0, 4: 0.6900,
            5: 0}, 'primary_task': 4},  # phi-4
        {'id': 22, 'intrinsic_competence': {1: 0, 2: 0, 3: 0, 4: 0.6600,
            5: 0}, 'primary_task': 4},  # Tulu-3-70B
        {'id': 23, 'intrinsic_competence': {1: 0, 2: 0, 3: 0, 4: 0.5800,
            5: 0}, 'primary_task': 4},  # Qwen2.5-Coder-7B-Inst
        {'id': 24, 'intrinsic_competence': {1: 0, 2: 0, 3: 0, 4: 0.5000,
            5: 0}, 'primary_task': 4},  # Llama-3.1-8B-Instruct
        # bbh_hyperbaton (task=5)
        {'id': 25, 'intrinsic_competence': {1: 0, 2: 0, 3: 0, 4: 0, 5:
            0.8100}, 'primary_task': 5},  # phi-4
        {'id': 26, 'intrinsic_competence': {1: 0, 2: 0, 3: 0, 4: 0, 5:
            0.7200}, 'primary_task': 5},  # Yi-1.5-34B-Chat
        {'id': 27, 'intrinsic_competence': {1: 0, 2: 0, 3: 0, 4: 0, 5:
            0.7100}, 'primary_task': 5},  # Llama-3.1-70B-Instruct
        {'id': 28, 'intrinsic_competence': {1: 0, 2: 0, 3: 0, 4: 0, 5:
            0.6700}, 'primary_task': 5},  # Tulu-3-70B
        {'id': 29, 'intrinsic_competence': {1: 0, 2: 0, 3: 0, 4: 0, 5:
            0.6800}, 'primary_task': 5},  # Llama-3.1-8B-Instruct
        {'id': 30, 'intrinsic_competence': {1: 0, 2: 0, 3: 0, 4: 0, 5:
            0.5200}, 'primary_task': 5},  # Qwen2.5-Coder-7B-Inst
    ],
    'edges': [
        (2, 3), (3, 4), (4, 5), (5, 6),
        (7, 8), (8, 9), (9, 10), (10, 11), (11, 12),
        (13, 14), (14, 15), (15, 16), (16, 17), (17, 18),
        (20, 21), (21, 22), (22, 23), (23, 24),
```

```
            (25, 26), (26, 27), (27, 28), (28, 29), (29, 30),
            (2, 7),  (6, 9),
            (5, 13), (4, 14),
            (4, 20), (2, 20),
            (3, 25), (6, 29),
            (10, 15), (8, 14),
            (8, 20), (7, 22),
            (10, 25), (9, 29),
            (14, 20), (13, 23),
            (15, 25), (18, 29),
            (21, 25), (20, 26),
        ],
    }
```

E LLM Agent Prompts

The following prompts were used for experiments characterizing the social intelligence of LLM agents presented in Fig. 3, with the same prompts used across all LLM agents.

Listing 1: System Prompt (Describing AgentSociety)

```
You are a node in a liquid democracy network making strategic decisions about voting
     and information sharing in order to maximise your payment (reward). YOU ARE A
    UTILITY MAXIMISER WITH YOUR UTILITY = PAYMENT. The game is defined as follows

### Tasks and Competence
- There is a single task to be completed
- Each node has ACTUAL COMPETENCE for the task (0.0 to 1.0 scale)
- Each node reports competence during delegaion based on which the payment (reward)
    received is computed. Each node is strategic here to report competence in such
    a way that the payment is maximized.
- Each node also diffuses a competence to each of its neighbours (could be different
     to different neighbours). Each node is strategic here because higher than
    required diffusion hurts its payment share, while lower than required might not
     get it on the winning path which would mean that there is no payment
    altogether.
- All agents maintain a log of what was diffused, so once a certain competence is
    diffused, not diffusing any information in the following iterations is the same
     as diffusing the previously diffused value as neighboring nodes log what was
    received from each node.
- The goal of each node is to maximize its payment and not just receive a payment.
- A positive payment means income and a negative payment means a penalty. Penalty is
     many times higher than income.
- As the diffusion happens between neighbors each iteration, there is a delay for
    the diffused value to reach extended neighbors in the graph to then influence
    them to change their voting based on diffused value.
- Therefore the agents need to consider that the reward and penalty can sometimes be
     lagging with respect to the information diffused.
- A rational agent starts with diffusing low values below its reported competence
    for payment and increases its diffusion value until required to maximize
    payment.
- A rational agent does not report competence for payment to be higher than max(
    actual competence, competence obtained from neighbors) on the primary task.

The system operates in the following steps -
The system broadcasts the task to all nodes.
The first step for each participating node is to vote, either for themselves or for
    one of their neighbors.
After the voting process concludes, votes aggregate at gurus, defined as agents who
    vote for themselves. Each guru represents a group of agents that have voted to
    it, with voting occurring transitively.
The voting paths are determined by the delegation paths formed by the transitive
    voting. The winning path is defined as the group with the highest number of
    agents.
Voting for a neighbor could be beneficial because there is finally only one winning
    path identified and only agents that make up this winning path have an
    opportunity to get paid.  The goal is to figure out who to vote for, given the
    payment mechanism in order to maximise own payment.
During voting, each node also sends the reported competence, that it can deliver
    either by itself or leveraging its neighbors by voting to them, to the system
    that will eventually be used for payment calculation.
Within this winning path, not all nodes are paid. Only the ones deemed critical, or
    those that would be part of a (different) winning path in case they had voted
    for themselves, are paid. These nodes make up the critical path sorted in the
    order of their competencies.
After identifying the critical path the framework assigns payments to utilizing the
    formula below by plugging their respective reported competence into it.
Let the critical path C_{t_k} be expressed as D_w^{t_k} = (a_1, a_2, ..., a_{k-1},
    a_k = w), which denotes the ordered sequence of agents whose vote forwarding
```

```
    was necessary for w to receive the allocation of task t_k. The payment function
     (positive payment means the agent gets paid) for an agent a_i in D_w^{t_k} is
    then given by p^(s_{sigma_j})(a_i) equals:

*   I(c_{i+1} >= c_i) * (f(c_i) - f(c_{i-1})), if a_i is in D_w^{t_k} excluding {w};
*   I(c_{i+1} >= c_i) * (f(c_i) - f(c_{i-1})), if a_i = w;
*   0, if a_i is not in D_w^{t_k},

where: c_i is the competence reported of agent a_i on task t_k, c_{i-1} = 0 if a_i =
     a_1, c_{i+1} = true competence of winner if a_i = w. I(.) is the indicator
    function, returning +1 if the condition holds and alpha(negative value)
    otherwise, f(.) is a monotone function modulating payments, and c is a baseline
     cost to the winning agent w to execute the task. Intuitively, the payment
    policy rewards or penalizes intermediaries according to the incremental
    competence they contribute along the vote chain.
After the payment is made, all agents are allowed to make a decision based on their
    current information state and payment outcome on whether they want to diffuse
    information to their neighbors to improve their own utility or payment.
```

Listing 2: Delegation Prompt

```
You are rational and intelligent Agent {node_id} in a system of all rational and
    intelligent agents making a delegation decision and reporting a competence for
    payment based on your delegation decision.

YOU NEED TO PAY ATTENTION AND UNDERSTAND TO THE PAYMENT FUNCTION IN ORDER TO
    MAXIMIZE YOUR PAYMENT for both your delegation decision as well as competence
    for payment.

## Your Information
- Your Primary Task: Task {primary_task}
- Your Actual Competence: {intrinsic_competence}
- Tasks Being Voted On: {tasks}

## Your Memory (History from Previous Iterations)
{memory_info}

## Your Neighbors competence (same primary task only)
{neighbors_info}

## Decision Required
You must choose  for Task {primary_task}, who to delegate your vote. Your options
    are:
1. Vote for yourself (delegate to yourself)
2. Delegate to one of your neighbors listed above (only those with primary task {
    primary_task})

If you choose to vote for yourself, use your own node ID ({node_id}).

Based on who you delegated to, report a corresponding competence which determines
    your payment.

Use your memory to learn from past delegation choices and their resulting payments.
You need to ensure that you delegate to a neighbor that at least can guarantee the
    performance you have already diffused. Otherwise your neighbors will no longer
    trust you to be accurate in your claims.

## Response Format
Respond with ONLY a JSON object. ALL three fields are REQUIRED and **DO NOT PROVIDE
    AN EMPTY RESPONSE**:
{{"delegate_to": <node_id>, "competence_for_payment": {{<task_id>: <competence_value
    >}}, "reasoning": "brief explanation"}}

Example response:
```

```
{{"delegate_to": 5, "competence_for_payment": {{"1": 0.7}}, "reasoning": "Node 5 has
     ..."}}

Field descriptions:
- delegate_to: The node ID you are delegating to (use your own ID {node_id} if
    voting for yourself)
- competence_for_payment: REQUIRED. A dictionary mapping each task_id (integer) to
    the competence value (float) you want to report for payment calculation.
    Include ALL tasks from {tasks}.
- reasoning: Brief explanation of your decision, keep it concise as there is a limit
     on the length
```

Listing 3: Diffusion Prompt

```
You are rational and intelligent Agent {node_id} in a system of all rational and
    intelligent agents deciding what information to diffuse to each of your
    neighbors.
Bear in mind that diffusion at this iteration can depend on the delegation decision
    of the same iteration.

## Your Information
- Your Primary Task: Task {primary_task}
- Your Actual Competence: {intrinsic_competence}
- Delegation Decision of current iteration: {delegated_to_info}
- Competence Reported for Payment (current iteration): {competence_for_payment_info}

## Your Memory (History from Previous Iterations)
{memory_info}

## Payment Outcome
{payment_status}

## Your Neighbors
- Same Primary Task (Task {primary_task}): {same_task_neighbors}
- Different Primary Tasks: {diff_task_neighbors}

## Neighbor Competence (what they have diffused to you)
Current State of Neighbors Competence: {neighbors_competence_info}

- You can strategically diffuse DIFFERENT competence values to DIFFERENT neighbors
  - Consider each neighbor's potential influence on maximizing your payment

- Use your memory to learn from past diffusion decisions and their impact on
    payments

## For example
If this is the received information -

# Your Information
- Your Primary Task: Task 1
- Your Actual Competence: {{"1": 0.2}}
- Delegation Decision of current iteration: You delegated to Node 5
- Competence Reported for Payment (current iteration): {{"1": 0.439}}

# Current State of Neighbors Competence: - Node 3 (Primary Task 1): Competence =
    {{"1": 0.419}}
- Node 5 (Primary Task 1): Competence = {{"1": 0.439}}

Here, diffusing a higher value of 0.425 to Node 3 would help you obtain node 3's
    vote while not losing substantial payment share. In this case, you would be
    acting as an intermediary transferring information from one node to another.


## Decision Required
```

```
Decide whether to diffuse information and what competence values to diffuse to EACH
    neighbor individually.

## Response Format
Respond with ONLY a JSON object in this exact format:
{{
  "should_diffuse": true/false,
  "neighbor_updates": {{
    "<neighbor_id>": {{"<task_id>": competence_value, ...}},
    "<neighbor_id>": {{"<task_id>": competence_value, ...}}
  }},
  "reasoning": "brief explanation of strategy"
}}

Example:
{{
  "should_diffuse": true,
  "neighbor_updates": {{
    "2": {{"1": 0.7}},
    "5": {{"1": 0.5}}
  }},
  "reasoning": "Reporting higher competence to node 2 because of ..."
}}

If should_diffuse is false, the updates will be ignored.
You must provide an update for each neighbor you want to diffuse information to.
```