



Multi-Agent Technologies: journey from Nash to LLM

Speaker: I. A. Akinfiev

SPbU
Saint-Petersburg, Russia
2025

Overview



The History of "Multi-Agent Systems" (Multi-Agency): This term historically described the function of multiple autonomous programs within distributed systems, later migrating for active use within the LLM domain.

The Feedback Loop: Ideas originating in the world of AI are now beginning to influence and shape the architecture of network services once again.

Conceptual Migration and Future Importance: Simple examples will demonstrate this cross-disciplinary transfer of concepts and explain why this pattern is critical for the future of computing systems.

Nash Equilibrium



Nash Equilibrium is a fundamental concept which describes a state in a strategic interaction between rational decision-makers where no player can improve their outcome by unilaterally changing their strategy.

Let a game be defined by the tuple $\Gamma = \langle I, S, u \rangle$.

Definitions:

1. **Players:** A finite set of players $I = \{1, 2, \dots, N\}$.
2. **Strategies:** For each player $i \in I$, there is a non-empty set of available strategies S_i .
 - $s_i \in S_i$: A specific strategy chosen by player i .
 - $s = (s_1, s_2, \dots, s_N)$: A **strategy profile** (the combination of choices made by all players).
 - $S = S_1 \times S_2 \times \dots \times S_N$: The set of all possible strategy profiles.
3. **Payoff (Utility):** A function $u_i : S \rightarrow \mathbb{R}$ representing the payoff for player i given a strategy profile.



Let (s_i, s_{-i}) denote a strategy profile where player i chooses s_i and all other players choose the strategies specified in the vector s_{-i} .

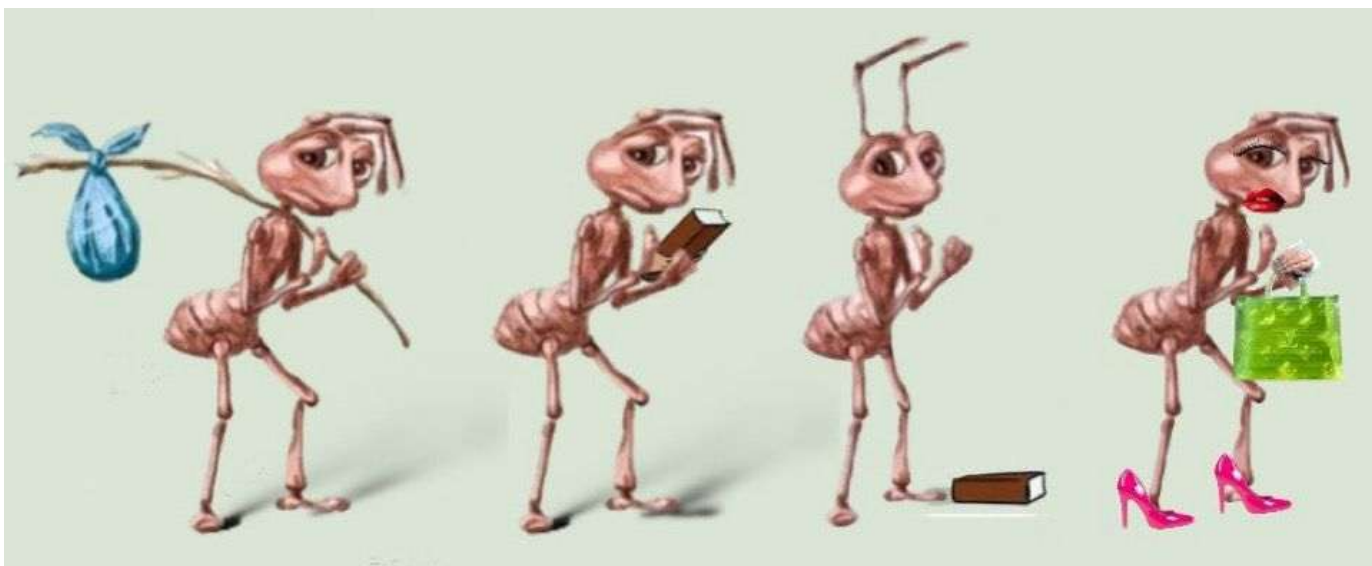
A strategy profile $s^* = (s_1^*, \dots, s_N^*)$ is a Nash Equilibrium if:

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*) \quad \forall s_i \in S_i, \quad \forall i \in I$$

In plain English: Given that all other players are playing their strategies from s^* , player i has no incentive to deviate from s_i^* to any other strategy s_i , because doing so would not increase their payoff.

A profile of mixed strategies $\sigma^* = (\sigma_1^*, \dots, \sigma_N^*)$ is a Nash Equilibrium if:

$$E[u_i(\sigma_i^*, \sigma_{-i}^*)] \geq E[u_i(\sigma_i, \sigma_{-i}^*)] \quad \forall \sigma_i \in \Sigma_i$$



SPbU
Saint-Petersburg, Russia
2025

Ant Colony Optimization



The translation of this biological behavior into computer code is credited to Marco Dorigo. In his PhD thesis at the Politecnico di Milano in 1992 (and in a technical report in 1991 with Maniezzo and Colorni), Dorigo proposed the Ant System (AS).

This was a paradigm shift. While Genetic Algorithms (which existed earlier) used populations, they were often treated as a pool of solutions manipulated by a central evolution process.

The Ant System was inherently distributed:

- Decentralized Control.
- Distributed Memory: the graph environment in the form of the pheromone matrix.



Given a set of cities and distances between them, find the shortest path that visits every city exactly once and returns to the start.

An ant k positioned at city i chooses to move to city j based on a probability rule. The probability P_{ij}^k is derived from the amount of pheromone on the edge and the heuristic desirability.

$$P_{ij}^k(t) = \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta_{ij}]^\beta}{\sum_{l \in N_i^k} [\tau_{il}(t)]^\alpha \cdot [\eta_{il}]^\beta}$$

Where:

- N_i^k : The set of cities ant k has not yet visited (feasible neighborhood).
- $\alpha \geq 0$: Parameter controlling the influence of the pheromone (history).
- $\beta \geq 1$: Parameter controlling the influence of the heuristic (greedy factor).

Let $G = (V, E)$ be a graph where V is the set of cities and E is the set of edges connecting them.

- d_{ij} : The distance between city i and city j .
- $\tau_{ij}(t)$: The intensity of the pheromone trail on edge (i, j) at time t .
- $\eta_{ij} = 1/d_{ij}$: The heuristic visibility (inversely proportional to distance; shorter edges are more attractive).

Ant Colony Optimization



When ant k is at city i , it chooses the next city j as follows:

$$j = \begin{cases} \arg \max_{l \in N_i^k} \{[\tau_{il}] \cdot [\eta_{il}]^\beta\}, & \text{if } q \leq q_0 \quad (\text{Exploitation}) \\ J, & \text{if } q > q_0 \quad (\text{Exploration}) \end{cases}$$

- q : A random number uniformly distributed in $[0, 1]$.
- q_0 : A tunable parameter.
- **If** $q \leq q_0$: The ant acts **deterministically** and chooses the absolute best edge available (Greedy approach).
- **If** $q > q_0$: The ant acts **probabilistically**, choosing variable J according to the standard random proportional rule (same as the original Ant System).

Consensus algorithm



The goal of the system is to minimize the **loss function**, defined as the average of local functionals:

$$J_n(x) = \frac{1}{N} \sum_{i=1}^N f_n^i(x) \rightarrow \min_x. \quad (3)$$

Consensus algorithm



$$J_n(x) = \frac{1}{N} \sum_{i=1}^N f_n^i(x) \rightarrow \min_x. \quad (3)$$

- decentralized operation systems
- live modeling
- time-synchronization demanding simulations at decentilized fog computing
- robotics applications (drone show, etc)
- other time depended distributed tasks

The ways the synchronization problem may be described:

- as a Consensus of distributed system
- as a Tracking problem

Communication network



Agents are able to communicate with each other through a network described by the undirected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where \mathcal{N} is a set of vertices and $\mathcal{E} \subseteq \mathcal{N} \times \mathcal{N}$ is a set of edges. Let $(j, i) \in \mathcal{E}$ if there is an edge between agents j and i . The latter means that agent j is send his current time estimation to agent i and vice versa. For an agent $i \in \mathcal{N}$, the set of neighbors is defined as $\mathcal{N}^i = \{j \in \mathcal{N} : (j, i) \in \mathcal{E}\}$. The *in-degree* of $i \in \mathcal{N}$ equals $|\mathcal{N}^i|$. Here and after, $|\cdot|$ is the cardinality of a set.

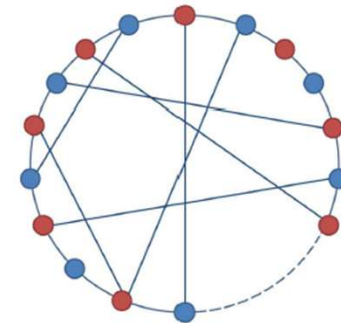


Figure 1: Topology example

Consensus algorithm



Find $w_k = 1 - \text{Argmin}_{X_k \in \mathbb{R}} \hat{X}_k(t), \forall_k = 1, \dots, k$

$$\omega_{k+1}(t) = 1 - \frac{1}{t_k^i h} \sum_{j \in N} b^{ij} (t_{k-1}^i + \frac{\omega_k}{\nu^i} - t_k^j),$$

where h is a step-size of consensus algorithm, t_k^j is time of j agent. Observation starts at $k = 0$ that makes algorithm applicable from $k = 1$ onwards.

Assumptions



Assumption 1. Functions F_k have a common Lipschitz constant $L > 0$ and strong convexity constant $\mu > 0$:

$$\forall \mathbf{t} \in \mathbb{R}^n \quad \|\nabla F_k(\mathbf{t})\| \leq L\|\mathbf{t} - \mathbf{t}_k\|, \quad (5)$$

$$\langle \nabla F_k(\mathbf{t}), \mathbf{t} - \mathbf{t}_k \rangle \geq \mu\|\mathbf{t} - \mathbf{t}_k\|^2. \quad (6)$$

Assumption 2. For every $k \geq 0$, drift is bounded as

$$\|F_k(\mathbf{t}) - F_{k+1}(\mathbf{t})\| \leq a\|\nabla F_k(\mathbf{t})\| + b, \quad (7)$$

$$\|\nabla F_{k+1}(\mathbf{t}) - \nabla F_k(\mathbf{t})\| \leq c. \quad (8)$$

Assumption 3. $\forall i \in \mathcal{N}, j \in \mathcal{N}^i$ the noises $\xi_k^{i,j}$ are centered and have bounded variance σ^2 .

Assumption 4. Communication graph \mathcal{G}_B is connected.

Tracking algorithm



1. Chose $\hat{t}_0^i \in \mathbb{R}$. (if time starts at 0 seconds chose 0). Set $t_o^i = \hat{t}_0^i$. Chose $h > 0, \eta \in (0, \mu), \alpha_x \in (0, 1)$ so that α_x satisfying the inequality (9) can always be found. Define $H_1 = h - \frac{h^2 L}{2}$.

2. k-th iteration ($k \geq 0$):

- (a) Find $\alpha_k \in [\alpha_x, 1)$ so that

$$H_1 - \frac{\alpha_k^2}{2\gamma_{k+1}} > 0.$$

- (b) Let $\gamma_{k+1} = (1 - \alpha_k)\gamma_k + \alpha_k(\mu - \eta)$.

- (c) Choose

$$z_k^i = \frac{1}{\gamma_k + \alpha_k(\mu - \eta)} (\alpha_k \gamma_k t_k^i + \gamma_{k+1} \hat{t}_k^i)$$

and find $f^i(z_k^i, t, \omega)$.

- (d) Find a new estimate $\hat{t}_{k+1}^i = z_k^i + h f^i(z_k^i, t, \omega)$.

- (e) Set

$$t_{k+1}^i = \frac{1}{\gamma_k} \left[(1 - \alpha_k) \gamma_k t_{k+1}^i + \alpha_k (\mu - \eta) z_k^i - \alpha_k f^i(z_k^i) \right].$$

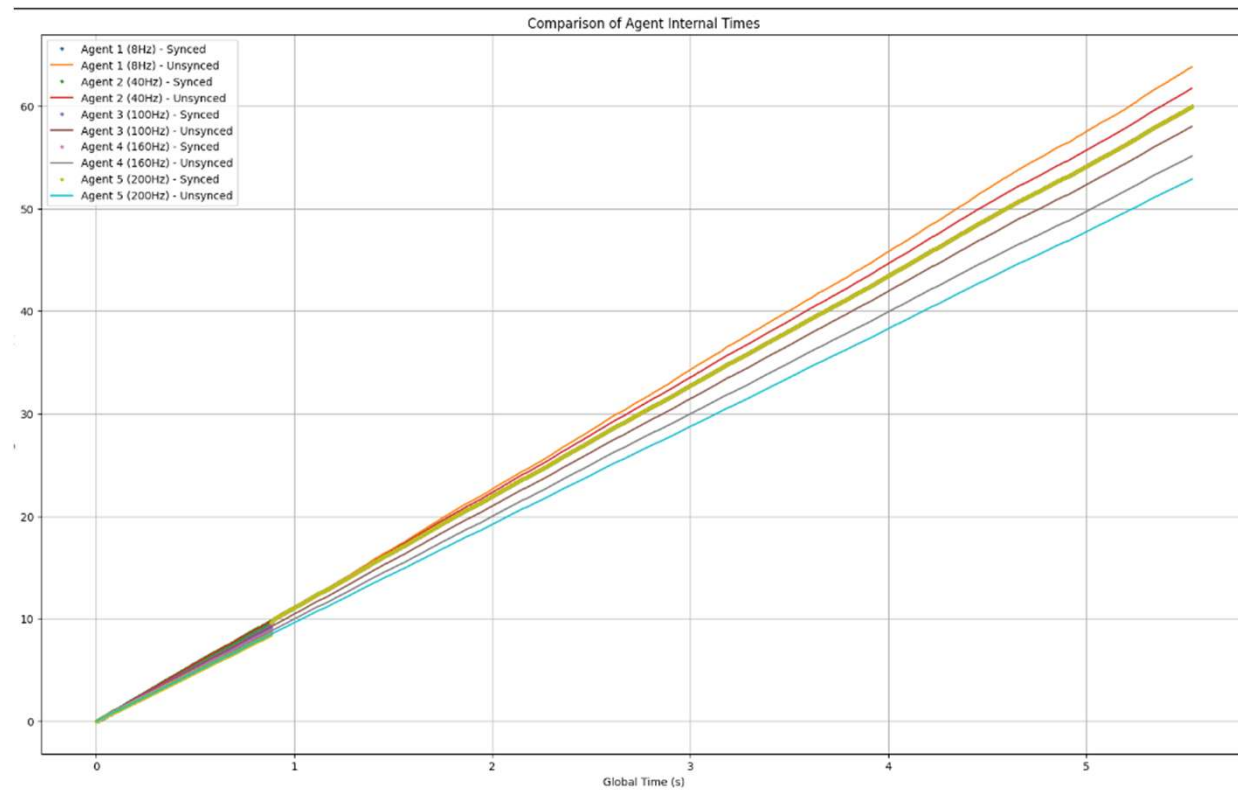
The proposed algorithm should be launched at each agent i independently.

D. Kosaty, A. Vakhitov, O. Granichin, and M. Yuchi, "Stochastic fast gradient for tracking," in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 1476–1481.

$$f(\hat{t}, t, \omega) = g \sum_{j \in N} (\hat{t}_k - t_k^j) \left(p_1 - \frac{p_1 g}{t_k^i} \sum_{j \in N} (\hat{t}_k - t_k^j) + p_3 - \frac{p_3}{t_k^i h} \sum_{j \in N} b^{ij} (t_{k-1}^i + \frac{\omega_k}{\nu^i} - t_k^j) \right),$$

where \hat{t} is a real time of a system, p_1, p_2, g are the weight coefficient of the function.

Simulation result example







1. Decentralized Training (Gossip Learning)

- **Concept:** Eliminates the central Parameter Server.
- **Mechanism:** Nodes exchange gradients or weights via **Average Consensus**.
- **Result:** The network mathematically converges to a global model using peer-to-peer communication.

2. Federated Fine-Tuning

- **Concept:** Private adaptation (e.g., LoRA) on edge devices.
- **Mechanism:** Consensus aggregates local weight updates while keeping raw data private.

3. Swarm Inference & Consistency

- **MechConcept:** Using multiple smaller models ("Mixture of Experts") instead of one giant model.
- **anism: Voting/Weighted Consensus** selects the most consistent answer among agent

Any questions?



SPbU
Saint-Petersburg, Russia
2025