Emergent Coordination in Multi-Agent Systems via Pressure Fields and Temporal Decay[1]

Roland R. Rodriguez, Jr.

Independent Researcher
rrrodzilla@proton.me

January 2026

Abstract. Current multi-agent Large Language Model (LLM) frameworks rely on explicit orchestration patterns borrowed from human organizational structures: planners delegate to executors, managers coordinate workers, and hierarchical control flow governs agent interactions. These approaches suffer from coordination overhead that scales poorly with agent count and task complexity. We propose a fundamentally different paradigm inspired by natural coordination mechanisms: agents operate locally on a shared artifact, guided only by pressure gradients derived from measurable quality signals, with temporal decay preventing premature convergence. We formalize this as optimization over a pressure landscape and prove convergence guarantees under mild conditions.

Empirically, on meeting room scheduling across 1350 trials, pressure-field coordination outperforms all baselines: 48.5% aggregate solve rate across all difficulty levels (86.7% on easy problems) versus 12.6% for conversation-based coordination, 1.5% for hierarchical control, and 0.4% for sequential and random baselines (all pairwise comparisons $p < 0.001$). Temporal decay is essential: disabling it reduces solve rate by 10 percentage points. On easy problems, pressure-field achieves 86.7% solve rate compared to 37.8% for the next-best baseline. The approach maintains consistent performance from 1 to 4 agents. Implicit coordination through shared pressure gradients outperforms explicit hierarchical control. Foundation models enable this approach: their broad pretraining and zero-shot reasoning allow quality-improving patches from local pressure signals alone, without domain-specific coordination protocols. This suggests that constraint-driven emergence offers a simpler and more effective foundation for multi-agent AI.

Keywords: multi-agent systems, emergent coordination, decentralized optimization, LLM agents

1 Introduction

Multi-agent systems built on large language models address complex task automation [Wu et al.(2023), Hong et al.(2023), Li et al.(2023)]. The dominant paradigm treats agents as organizational units: planners decompose tasks, managers delegate subtasks, and workers execute instructions under hierarchical supervision. This coordination overhead scales poorly with agent count and task complexity.

We demonstrate that *implicit* coordination through shared state outperforms explicit hierarchical control—without coordinators, planners, or message passing. Across 1350 trials on meeting room scheduling, pressure-field coordination achieves 48.5% aggregate solve rate compared to 1.5% for hierarchical control ($p < 0.001$, Cohen's $h = 1.97$ on easy problems). Conversation-based coordination (AutoGen-style) achieves 12.6%, while sequential

---

[1] Code available at https://github.com/Govcraft/pressure-field-experiment

and random baselines achieve only 0.4%.

Our approach draws inspiration from natural coordination mechanisms—ant colonies, immune systems, neural tissue—that coordinate through *environment modification* rather than message passing. Agents observe local quality signals (pressure gradients), take locally-greedy actions, and coordination emerges from shared artifact state. The key insight is that *local greedy decisions are effective for constraint satisfaction*: when problems exhibit locality (fixing one region rarely breaks distant regions), decentralized greedy optimization outperforms centralized planning. Temporal decay prevents premature convergence by ensuring continued exploration.

Our contributions:

1. We formalize *pressure-field coordination* as a role-free, stigmergic alternative to organizational Multi-Agent System (MAS) paradigms. Unlike Generalized Partial Global Planning (GPGP)'s hierarchical message-passing or SharedPlans' intention alignment, pressure-field achieves $O(1)$ coordination overhead through shared artifact state. Foundation models enable this approach: their broad pretraining allows quality-improving patches from local pressure signals without domain-specific coordination protocols.

2. We introduce *temporal decay* as a mechanism for preventing premature convergence. Disabling decay reduces solve rate by 10 percentage points (from 96.7% to 86.7% in ablation studies), trapping agents in local minima.

3. We prove convergence guarantees for this coordination scheme under pressure alignment conditions.

4. We provide empirical evidence across 1350 trials showing: (a) pressure-field outperforms hierarchical control (48.5% vs 1.5%), (b) all comparisons with baselines are highly significant ($p < 0.001$).

2 Related Work

Our approach bridges four research traditions: multi-agent systems coordination theory provides the conceptual foundation; swarm intelligence provides the stigmergic mechanism; LLM systems provide the application domain; and decentralized optimization provides theoretical guarantees. We survey each and position pressure-field coordination within this landscape.

2.1 MAS Coordination Theory

Pressure-field coordination occupies a unique position in the MAS landscape: it eliminates roles (unlike organizational paradigms), messages (unlike GPGP), and intention reasoning (unlike SharedPlans) while providing formal convergence guarantees (unlike purely reactive systems). This section positions our contribution within four established coordination frameworks, showing how artifact refinement with measurable quality signals enables this architectural simplification. For this domain class, coordination complexity collapses from quadratic message-passing to constant-time state-sharing.

2.1.1 Organizational Paradigms and Dependency Management

Pressure-field coordination achieves role-free coordination: any agent can address any high-pressure region without negotiating access rights or awaiting task assignment. This contrasts sharply with traditional organizational paradigms. Horling and Lesser [Horling and Lesser(2004)] surveyed nine such paradigms—from rigid hierarchies to flexible markets—finding that all assign explicit roles constraining agent behavior. While role assignment reduces coordination complexity by pre-structuring interactions, it introduces brittleness: role changes require protocol modifications, and role failure can cascade through the system.

Our approach instantiates Malone and Crowston's [Malone and Crowston(1994)] coordi-

nation framework with a critical difference: the artifact itself is the shared resource, and pressure gradients serve as dependency signals. Rather than assigning roles to manage resource access, agents share read access to the entire artifact and propose changes to high-pressure regions. Coordination emerges from pressure alignment—agents reduce local pressure, which reduces global pressure through the artifact's shared state.

2.1.2 Distributed Problem Solving and Communication Overhead

Pressure-field coordination achieves $O(1)$ inter-agent communication overhead—agents exchange no messages. Coordination occurs entirely through shared artifact reads and writes, eliminating the message-passing bottleneck. This contrasts with the GPGP framework [Decker and Lesser(1995)], which reduces communication from $O(n^2)$ pairwise negotiation to $O(n \log n)$ hierarchical aggregation through summary information exchange. While GPGP represents significant progress, its explicit messages—task announcements, commitment exchanges, schedule updates—still introduce latency and failure points at scale.

The approaches target different domains. Pressure-field coordination specializes in artifact refinement tasks where quality decomposes into measurable regional signals—a class including code quality improvement, document editing, and configuration management. GPGP generalizes to complex task networks with precedence constraints. For artifact refinement, however, pressure-field's stigmergic coordination eliminates message-passing overhead entirely.

2.1.3 Shared Intentions and Alignment Costs

Pressure-field coordination eliminates intention alignment through pressure alignment. Rather than reasoning about what other agents believe or intend, agents observe artifact state and pressure gradients. When agents greedily reduce local pressure under separable or bounded-coupling conditions, global pressure decreases. This is coordination without communication about intentions—agents align through shared objective functions, not mutual beliefs.

This contrasts with the SharedPlans framework [Grosz and Kraus(1996)], which formalizes joint activity through shared mental attitudes: mutual beliefs about goals, commitments, and action sequences. The framework captures human-like collaboration but requires significant cognitive machinery—intention recognition, commitment protocols, belief revision—all computationally expensive operations that scale poorly with agent count.

Our experiments validate this analysis: pressure-field coordination eliminates the overhead of explicit dialogue by coordinating through shared artifact state. The coordination overhead of belief negotiation in explicit dialogue systems can exceed its organizational benefit for constraint satisfaction tasks. The trade-off is transparency: SharedPlans supports dialogue about why agents act; pressure-field agents react to gradients without explaining reasoning.

2.1.4 Self-Organization and Emergent Coordination

Pressure-field coordination satisfies De Wolf and Holvoet's [De Wolf and Holvoet(2005)] self-organization criteria: absence of external control, local interactions producing global patterns, and dynamic adaptation. They explicitly cite "gradient fields" as a self-organization design pattern—our approach instantiates this pattern with formal guarantees.

No external controller exists—agents observe and act autonomously based on local pressure signals. Coordination emerges from local decisions: agents reduce regional pressure through greedy actions, and global coordination arises from shared artifact state. Temporal decay provides dynamic adaptation—fitness erodes continuously, preventing premature convergence and enabling continued refinement.

The theoretical contribution formalizes this intuition through potential game theory. Theorem 1 establishes convergence guarantees for aligned pressure systems; the Basin Separation result (Theorem 3) explains why decay is necessary to escape suboptimal basins.

2.1.5 Foundation Model Enablement

Foundation Models (FMs) enable stigmergic coordination through three capabilities: (1) broad pretraining allows patch proposals across diverse artifact types without domain-specific fine-tuning; (2) instruction-following allows operation from pressure signals alone, without complex action representations; (3) zero-shot reasoning interprets constraint violations without explicit protocol training. These properties make FMs suitable for stigmergic coordination—they require only local context and quality signals to generate productive actions, matching pressure-field's locality constraints.

2.2 Multi-Agent LLM Systems

Recent work has explored multi-agent architectures for LLM-based task solving. AutoGen [Wu et al.(2023)] introduces a conversation-based framework where customizable agents interact through message passing, with support for human-in-the-loop workflows. MetaGPT [Hong et al.(2023)] encodes Standardized Operating Procedures (SOPs) into agent workflows, assigning specialized roles (architect, engineer, QA) in an assembly-line paradigm. CAMEL [Li et al.(2023)] proposes role-playing between AI assistant and AI user agents, using inception prompting to guide autonomous cooperation. CrewAI [CrewAI(2024)] similarly defines agents with roles, goals, and backstories that collaborate on complex tasks.

These frameworks share a common design pattern: explicit orchestration through message passing, role assignment, and hierarchical task decomposition. While effective for structured workflows, this approach faces scaling limitations. Central coordinators become bottlenecks, message-passing overhead grows with agent count, and failures in manager agents cascade to dependents. Our work takes a fundamentally different approach: coordination emerges from shared state rather than explicit communication.

Foundation models enable pressure-field coordination through capabilities that prior agent architectures lacked. Their broad pretraining allows patches across diverse artifact types—code, text, configurations—without domain-specific fine-tuning. Their instruction-following capabilities allow operation from pressure signals and quality feedback alone. Their zero-shot reasoning interprets constraint violations and proposes repairs without explicit protocol training. These properties make foundation models particularly suitable for stigmergic coordination: they require only local context and quality signals to generate productive actions, matching the locality constraints of pressure-field systems.

2.3 Swarm Intelligence and Stigmergy

The concept of stigmergy—indirect coordination through environment modification—was introduced by Grassé [Grassé(1959)] to explain termite nest-building behavior. Termites deposit pheromone-infused material that attracts further deposits, leading to emergent construction without central planning. This directly instantiates Malone and Crowston's [Malone and Crowston(1994)] shared resource coordination: pheromone trails encode dependency information about solution quality. Complex structures arise from simple local rules without any agent having global knowledge.

Dorigo and colleagues [Dorigo et al.(1996), Dorigo and Gambardella(1997)] formalized this insight into Ant Colony Optimization (ACO), where artificial pheromone trails guide search through solution spaces. Key mechanisms include positive feedback (reinforcing good paths), negative feedback (pheromone evaporation), and purely local decision-making. ACO has achieved strong results on combinatorial optimization problems including Traveling Salesman Problem (TSP), vehicle routing, and scheduling.

Our pressure-field coordination directly inherits from stigmergic principles. The artifact serves as the shared environment; regional pressures are analogous to pheromone concentrations; decay corresponds to evaporation. However, we generalize beyond path-finding to arbitrary artifact refinement and provide formal convergence guarantees through the potential game framework.

2.4 Decentralized Optimization

Potential games, introduced by Monderer and Shapley [Monderer and Shapley(1996)], are games where individual incentives align with a global potential function. A key property is that any sequence of unilateral improvements converges to a Nash equilibrium—greedy local play achieves global coordination. This provides the theoretical foundation for our convergence guarantees: under pressure alignment, the artifact pressure serves as a potential function.

Distributed gradient descent methods [Nedić and Ozdaglar(2009), Yuan et al.(2016)] address optimization when data or computation is distributed across nodes. The standard approach combines local gradient steps with consensus averaging. While these methods achieve convergence rates matching centralized alternatives, they typically require communication protocols and synchronization. Our approach avoids explicit communication entirely: agents coordinate only through the shared artifact, achieving $O(1)$ coordination overhead.

The connection between multi-agent learning and game theory has been extensively studied [Shoham and Leyton-Brown(2008)]. Our contribution is applying these insights to LLM-based artifact refinement, where the "game" is defined by pressure functions over quality signals rather than explicit reward structures.

3 Problem Formulation

We formalize artifact refinement as a dynamical system over a pressure landscape rather than an optimization problem with a target state. The system evolves through local actions and continuous decay, settling into stable basins that represent acceptable artifact states.

3.1 State Space

An *artifact* consists of n regions with content $c_i \in \mathcal{C}$ for $i \in \{1, \ldots, n\}$, where $\mathcal{C}$ is an arbitrary content space (strings, Abstract Syntax Tree (AST) nodes, etc.). Each region also carries auxiliary state $h_i \in \mathcal{H}$ representing confidence, fitness, and history. Regions are passive subdivisions of the artifact; agents are active proposers that observe regions and generate patches.

The full system state is:

$$s = ((c_1, h_1), \ldots, (c_n, h_n)) \in (\mathcal{C} \times \mathcal{H})^n$$

3.2 Pressure Landscape

A *signal function* $\sigma : \mathcal{C} \to \mathbb{R}^d$ maps content to measurable features. Signals are *local*: $\sigma(c_i)$ depends only on region i.

A *pressure function* $\phi : \mathbb{R}^d \to \mathbb{R}_{\geq 0}$ maps signals to scalar "badness." We consider k pressure axes with weights $\mathbf{w} \in \mathbb{R}^k_{>0}$. The *region pressure* is:

$$P_i(s) = \sum_{j=1}^{k} w_j \phi_j(\sigma(c_i))$$

The *artifact pressure* is:

$$P(s) = \sum_{i=1}^{n} P_i(s)$$

This defines a landscape over artifact states. Low-pressure regions are "valleys" where the artifact satisfies quality constraints.

3.3 System Dynamics

The system evolves in discrete time steps (ticks). Each tick consists of four phases:

Phase 1: Decay. Auxiliary state erodes toward a baseline. For fitness f_i and confidence γ_i components of h_i:

$$f_i^{t+1} = f_i^t \cdot e^{-\lambda_f}, \quad \gamma_i^{t+1} = \gamma_i^t \cdot e^{-\lambda_\gamma}$$

where $\lambda_f, \lambda_\gamma > 0$ are decay rates. Decay ensures that stability requires continuous reinforcement.

Phase 2: Proposal. For each region i where pressure exceeds activation threshold ($P_i > \tau_{\text{act}}$) and the region is not inhibited, *each actor $a_k : \mathcal{C} \times \mathcal{H} \times \mathbb{R}^d \to \mathcal{C}$ proposes a content transformation in parallel*. Each actor observes only local state $(c_i, h_i, \sigma(c_i))$—actors do not communicate or coordinate their proposals.

Phase 3: Validation. When multiple patches are proposed, each is validated on an independent *fork* of the artifact. Forks are created by cloning artifact state; validation proceeds in parallel across forks. This addresses a fundamental resource constraint: a single artifact cannot be used to test multiple patches simultaneously without cloning.

Phase 4: Reinforcement. Regions where actions were applied receive fitness and confidence boosts, and enter an inhibition period preventing immediate re-modification. Inhibition allows changes to propagate through the artifact and forces agents to address other high-pressure regions, preventing oscillation around local fixes.

$$f_i^{t+1} = \min(f_i^t + \Delta_f, 1), \quad \gamma_i^{t+1} = \min(\gamma_i^t + \Delta_\gamma, 1)$$

3.4 Stable Basins

Definition 1 (Stability). A state s^* is *stable* if, under the system dynamics with no external perturbation:

1. All region pressures are below activation threshold: $P_i(s^*) < \tau_{\text{act}}$ for all i

2. Decay is balanced by residual fitness: the system remains in a neighborhood of s^*

The central questions are:

1. **Existence**: Under what conditions do stable basins exist?

2. **Quality**: What is the pressure $P(s^*)$ of states in stable basins?

3. **Convergence**: From initial state s_0, does the system reach a stable basin? How quickly?

4. **Decentralization**: Can stability be achieved with purely local decisions?

3.5 The Locality Constraint

The constraint distinguishing our setting from centralized optimization: agents observe only local state. An actor at region i sees $(c_i, h_i, \sigma(c_i))$ but not:

- Other regions' content c_j for $j \neq i$

- Global pressure $P(s)$

- Other agents' actions

This rules out coordinated planning. Stability must emerge from local incentives aligned with global pressure reduction.

4 Method

We now present a coordination mechanism that achieves stability through purely local decisions. Under appropriate conditions, the artifact pressure $P(s)$ acts as a *potential function*: local improvements by individual agents decrease global pressure, guaranteeing convergence without coordination.

4.1 Pressure Alignment

The locality constraint prohibits agents from observing global state. For decentralized coordination to succeed, we need local incentives to align with global pressure reduction.

Definition 2 (Pressure Alignment). A pressure system is *aligned* if for any region i, state s, and action a_i that reduces local pressure:

$$P_i(s') < P_i(s) \implies P(s') < P(s)$$

where $s' = s[c_i \mapsto a_i(c_i)]$ is the state after applying a_i.

Alignment holds automatically when pressure functions are *separable*: each P_i depends only on c_i, so $P(s) = \sum_i P_i(s)$ and local improvement directly implies global improvement.

More generally, alignment holds when cross-region interactions are bounded:

Definition 3 (Bounded Coupling). A pressure system has ϵ-*bounded coupling* if for any action a_i on region i:

$$|P_j(s') - P_j(s)| \leq \epsilon \quad \forall j \neq i$$

That is, modifying region i changes other regions' pressures by at most ϵ.

Under ϵ-bounded coupling with n regions, if a local action reduces P_i by $\delta > (n-1)\epsilon$, then global pressure decreases by at least $\delta - (n-1)\epsilon > 0$.

4.2 Connection to Potential Games

The aligned pressure system forms a *potential game* where:

- Players are regions (or agents acting on regions)
- Strategies are content choices $c_i \in \mathcal{C}$
- The potential function is $\Phi(s) = P(s)$

In potential games, any sequence of improving moves converges to a Nash equilibrium. In our setting, Nash equilibria correspond to stable basins: states where no local action can reduce pressure below the activation threshold.

This connection provides our convergence guarantee without requiring explicit coordination.

Note that this convergence result assumes finite action spaces. In practice, patches are drawn from a finite set of LLM-generated proposals per region, satisfying this requirement. For infinite content spaces, convergence to approximate equilibria can be established under Lipschitz continuity conditions on pressure functions.

4.3 The Coordination Algorithm

The tick loop implements greedy local improvement with decay-driven exploration:

The algorithm has three key properties:

Locality. Each actor observes only $(c_i, h_i, \sigma(c_i))$. No global state is accessed.

Bounded parallelism. At most κ patches per tick prevents thrashing. Inhibition prevents repeated modification of the same region.

Decay-driven exploration. Even stable regions eventually decay below confidence thresholds, attracting re-evaluation. This prevents premature convergence to local minima.

4.4 Stability and Termination

The system reaches a stable basin when:

1. All region pressures satisfy $P_i(s) < \tau_{\text{act}}$
2. Decay is balanced: fitness remains above the threshold needed for stability

---

Algorithm 1: **Pressure-Field Tick**

Input: State s^t, signal functions $\{\sigma_j\}$, pressure functions $\{\phi_j\}$, actors $\{a_k\}$, parameters $(\tau_{\text{act}}, \lambda_f, \lambda_\gamma, \Delta_f, \Delta_\gamma, \kappa)$

Phase 1: Decay

For each region i: $\quad f_i \leftarrow f_i \cdot e^{-\lambda_f}, \quad \gamma_i \leftarrow \gamma_i \cdot e^{-\lambda_\gamma}$

Phase 2: Activation and Proposal

$\mathcal{P} \leftarrow \emptyset$

For each region i where $P_i(s) \geq \tau_{\text{act}}$ and not inhibited:

$\boldsymbol{\sigma}_i \leftarrow \sigma(c_i)$

For each actor a_k:

$\delta \leftarrow a_k(c_i, h_i, \boldsymbol{\sigma}_i)$

$\mathcal{P} \leftarrow \mathcal{P} \cup \{(i, \delta, \hat{\Delta}(\delta))\}$

Phase 3: Parallel Validation and Selection

For each candidate patch $(i, \delta, \hat{\Delta}) \in \mathcal{P}$:

Fork artifact: $(f_{\text{id}}, A_f) \leftarrow A.\text{fork}()$

Apply δ to fork A_f

Validate fork (run tests, check compilation)

Collect validation results $\{(i, \delta, \Delta_{\text{actual}}, \text{valid})\}$

Sort validated patches by Δ_{actual}

Greedily select top-κ non-conflicting patches

Phase 4: Application and Reinforcement

For each selected patch $(i, \delta, \cdot)$:

$c_i \leftarrow \delta(c_i)$

$f_i \leftarrow \min(f_i + \Delta_f, 1), \gamma_i \leftarrow \min(\gamma_i + \Delta_\gamma, 1)$

Mark region i inhibited for τ_{inh} ticks

Return updated state s^{t+1}

Termination is *economic*, not logical. The system stops acting when the cost of action (measured in pressure reduction per patch) falls below the benefit. This matches natural systems: activity ceases when gradients flatten, not when an external goal is declared achieved.

In practice, we also impose budget constraints (maximum ticks or patches) to bound computation.

5 Theoretical Analysis

We establish three main results: (1) convergence to stable basins under alignment, (2) bounds on stable basin quality, and (3) scaling properties relative to centralized alternatives.

5.1 Convergence Under Alignment

Theorem 1 (Convergence). Let the pressure system be aligned with ϵ-bounded coupling. Let $\delta_{\min} > 0$ be the minimum *local* pressure reduction $P_i(s) - P_i(s')$ from any applied patch, and assume $\delta_{\min} > (n-1)\epsilon$ where n is the number of regions. Then from any initial state s_0 with pressure $P_0 = P(s_0)$, the system reaches a stable basin within:

$$T \leq \frac{P_0}{\delta_{\min} - (n-1)\epsilon}$$

ticks, provided the fitness boost Δ_f from successful patches exceeds decay during inhibition: $\Delta_f > 1 - e^{-\lambda_f \cdot \tau_{\mathrm{inh}}}$.

Proof sketch. Under alignment with ϵ-bounded coupling, each applied patch reduces global pressure by at least $\delta_{\min} - (n-1)\epsilon > 0$. Since $P(s) \geq 0$ and decreases by a fixed minimum per tick (when patches are applied), the system must reach a state where no region exceeds τ_{act} within the stated bound. The decay constraint ensures that stability is maintained once reached: fitness reinforcement from the final patches persists longer than the decay erodes it. □

The bound is loose but establishes that convergence time scales with initial pressure, not with state space size or number of possible actions.

5.2 Basin Quality

Theorem 2 (Basin Quality). In any stable basin s^*, the artifact pressure satisfies:

$$P(s^*) < n \cdot \tau_{\mathrm{act}}$$

where n is the number of regions and τ_{act} is the activation threshold.

Proof. By definition of stability, $P_i(s^*) < \tau_{\mathrm{act}}$ for all i. Summing over regions: $P(s^*) = \sum_i P_i(s^*) < n \cdot \tau_{\mathrm{act}}$. □

This bound is tight: adversarial initial conditions can place the system in a basin where each region has pressure just below threshold. However, in practice, actors typically reduce pressure well below τ_{act}, yielding much lower basin pressures.

Theorem 3 (Basin Separation). Under separable pressure (zero coupling), distinct stable basins are separated by pressure barriers of height at least τ_{act}.

Proof sketch. Moving from one basin to another requires some region to exceed τ_{act} (otherwise no action is triggered). The minimum such exceedance defines the barrier height. □

This explains why decay is necessary: without decay, the system can become trapped in suboptimal basins. Decay gradually erodes fitness, eventually allowing re-evaluation and potential escape to lower-pressure basins.

5.3 Scaling Properties

Theorem 4 (Linear Scaling). Let m be the number of regions and n be the number of parallel agents. The per-tick complexity is:

- **Signal computation:** $O(m \cdot d)$ where d is signal dimension
- **Pressure computation:** $O(m \cdot k)$ where k is the number of pressure axes
- **Patch proposal:** $O(m \cdot a)$ where a is the number of actors
- **Selection:** $O(m \cdot a \cdot \log(m \cdot a))$ for sorting candidates
- **Coordination overhead:** $O(1)$—no inter-agent communication (fork pool is $O(K)$ where K is fixed)

Total: $O(m \cdot (d + k + a \cdot \log(ma)))$, independent of agent count n.

Adding agents increases throughput (more patches proposed per tick) without increasing coordination cost. This contrasts with hierarchical schemes where coordination overhead grows with agent count.

Theorem 5 (Parallel Convergence). Under the same alignment conditions as Theorem 1, with K patches validated in parallel per tick where patches affect disjoint regions, the system reaches a stable basin within:

$$T \leq \frac{P_0}{K \cdot (\delta_{\min} - (n-1)\epsilon)}$$

This improves convergence time by factor K while maintaining guarantees.

Proof sketch. When K non-conflicting patches are applied per tick, each reduces global pressure by at least $\delta_{\min} - (n-1)\epsilon$. The combined reduction is $K \cdot (\delta_{\min} - (n-1)\epsilon)$ per tick. The bound follows directly. Note that if patches conflict (target the same region), only one is selected per region, and effective speedup is reduced. □

5.4 Comparison to Alternatives

We compare against three coordination paradigms:

Centralized planning. A global planner evaluates all $(m \cdot a)$ possible actions, selects optimal subset. Per-step complexity: $O(m \cdot a)$ evaluations, but requires global state access. Sequential bottleneck prevents parallelization.

Hierarchical delegation. Manager agents decompose tasks, delegate to workers. Communication complexity: $O(n \log n)$ for tree-structured delegation with n agents. Latency scales with tree depth. Failure of manager blocks all descendants.

Message-passing coordination. Agents negotiate actions through pairwise communication. Convergence requires $O(n^2)$ messages in worst case for n agents. Consensus protocols add latency.

Paradigm	Coordination	Parallelism	Fault tolerance
Centralized	$O(m \cdot a)$	None	Single point of failure
Hierarchical	$O(n \log n)$	Limited by tree	Manager failure cascades
Message-passing	$O(n^2)$	Consensus-bound	Partition-sensitive
Pressure-field	$O(1)$	Full $(\min(n, m, K))$	Graceful degradation

Table 1: Coordination overhead comparison. K denotes the fork pool size for parallel validation.

Pressure-field coordination achieves $O(1)$ coordination overhead because agents share state only through the artifact itself—a form of stigmergy. Agents can fail, join, or leave without protocol overhead.

6 Experiments

We evaluate pressure-field coordination on meeting room scheduling: assigning N meetings to R rooms over D days to minimize gaps (unscheduled time), overlaps (attendee double-bookings), and maximize utilization balance. This domain provides continuous pressure gradients (rather than discrete violations), measurable success criteria, and scalable difficulty through problem size.

Key findings: Pressure-field coordination outperforms all baselines (§6.2). Temporal decay is critical—disabling it reduces solve rate by 10 percentage points (§6.3). The approach maintains consistent performance from 1 to 4 agents (§6.4).

6.1 Setup

6.1.1 Task: Meeting Room Scheduling

We generate scheduling problems with varying difficulty:

Difficulty	Rooms	Meetings	Pre-scheduled
Easy	3	20	70%
Medium	5	40	50%
Hard	5	60	30%

Table 2: Problem configurations. Pre-scheduled percentage indicates meetings already placed; remaining must be scheduled by agents.

Each schedule spans 5 days with 30-minute time slots (8am–4pm). Regions are 2-hour time blocks (4 blocks per day × 5 days = 20 regions per schedule). A problem is "solved" when all meetings are scheduled with zero attendee overlaps within 50 ticks.

Pressure function: $P = \text{gaps} \cdot 1.0 + \text{overlaps} \cdot 2.0 + \text{util\_var} \cdot 0.5 + \text{unsched} \cdot 1.5$

where gaps measures empty slots as a fraction, overlaps counts attendee double-bookings, util_var measures room utilization variance, and unsched is the fraction of unscheduled meetings.

6.1.2 Baselines

We compare five coordination strategies, all using identical LLMs (`qwen2.5:0.5b/1.5b/3b` via Ollama) to isolate coordination effects:

Pressure-field (ours): Full system with decay (fitness half-life 5s), inhibition (2s cooldown), greedy region selection (highest-pressure region per tick), and parallel validation. Includes band escalation (Exploitation → Balanced → Exploration) and model escalation (0.5b → 1.5b → 3b).

Conversation: AutoGen-style multi-agent dialogue where agents exchange messages to coordinate scheduling decisions. Agents discuss conflicts and propose solutions through explicit communication.

Hierarchical: Single agent selects the highest-pressure time block each tick, proposes a schedule change, and validates before applying (only accepts pressure-reducing patches). Uses identical prompts to pressure-field. The differences are: (1) greedy region selection always targets the hardest region, and (2) sequential execution processes one region per tick. This represents centralized, quality-gated control.

Sequential: Single agent iterates through time blocks in fixed order, proposing schedule changes one region at a time. No parallelism, pressure guidance, or patch validation—applies any syntactically valid patch regardless of quality impact.

Random: Selects random time blocks and proposes schedule changes. No patch validation—applies any syntactically valid patch regardless of quality impact.

Note on parallelism: Pressure-field validates multiple patches in parallel (K regions per tick), while hierarchical validates one patch sequentially. This asymmetry is *inherent to the coordination paradigm*, not an implementation choice: hierarchical control requires the manager to select a region, delegate to a worker, and validate the result before proceeding— delegating to multiple workers simultaneously would require additional coordination protocols (work distribution, conflict resolution, result aggregation) that would transform it into a different architecture entirely. The sequential bottleneck is the cost of centralized control. When hierarchical's single patch is rejected, the tick produces no progress; when one of pressure-field's parallel patches is rejected, others may still succeed.

Model choice rationale: We deliberately use small, minimally-capable models (0.5b–3b parameters) rather than frontier models. This design choice strengthens our thesis: if the coordination mechanism can extract effective performance from weak models, the mechanism itself is valuable—independent of model capability. Using identical model chains across all strategies isolates coordination effects from model effects.

6.1.3 Metrics

- **Solve rate**: Percentage of schedules reaching all meetings placed with zero overlaps within 50 ticks.

- **Ticks to solve**: Convergence speed for solved cases

- **Final pressure**: Remaining gaps, overlaps, and unscheduled meetings for unsolved cases

6.1.4 Implementation

Hardware: NVIDIA RTX 4070 8GB Graphics Processing Unit (GPU), AMD Ryzen 9 7940HS, 64GB RAM. **Software**: Rust implementation with Ollama. **Trials**: 30 per configuration. Full protocol in Appendix A.

Band escalation: When pressure velocity (rate of improvement) drops to zero for 7 consecutive ticks, sampling parameters escalate: Exploitation (T=0.2, p=0.85) $\rightarrow$ Balanced (T=0.4, p=0.9) $\rightarrow$ Exploration (T=0.7, p=0.95).

Model escalation: After exhausting all bands with zero progress (21 ticks total), the system escalates through the model chain: 0.5b $\rightarrow$ 1.5b $\rightarrow$ 3b, resetting to Exploitation band. Section 6.5 analyzes this mechanism.

6.2 Main Results

Across 1350 total trials spanning three difficulty levels (easy, medium, hard) and agent counts (1, 2, 4), we find that pressure-field coordination outperforms all baselines:

Strategy	Solved/N	Rate	95% Wilson CI
Pressure-field	131/270	48.5%	42.6%–54.5%
Conversation	34/270	12.6%	9.2%–17.1%
Hierarchical	4/270	1.5%	0.6%–3.7%
Sequential	1/270	0.4%	0.1%–2.1%
Random	1/270	0.4%	0.1%–2.1%

Table 3: Aggregate solve rates across all experiments (1350 total trials, 270 per strategy). Chi-square test across all five strategies: $\chi^2 > 200$, $p < 0.001$.

The results show clear stratification:

Pressure-field dominates: Pressure-field achieves 48.5% solve rate, nearly 4$\times$ higher than the next-best baseline (conversation at 12.6%). The effect size is large: Cohen's $h = 1.07$ versus conversation on easy problems, and $h > 1.97$ versus all other baselines.

Conversation provides intermediate performance: The AutoGen-style conversa-

tion baseline achieves 12.6% overall, significantly better than hierarchical ($p < 0.001$) but far below pressure-field. Notably, conversation solves only easy problems (37.8% on easy, 0% on medium and hard).

Hierarchical and sequential fail: Despite explicit coordination, hierarchical control achieves only 1.5% solve rate—comparable to random (0.4%). Both strategies fail entirely on medium and hard problems.

This result contradicts the common assumption that explicit hierarchical coordination should outperform implicit coordination. The overhead of centralized control and message passing appears to harm rather than help performance on constraint satisfaction tasks.
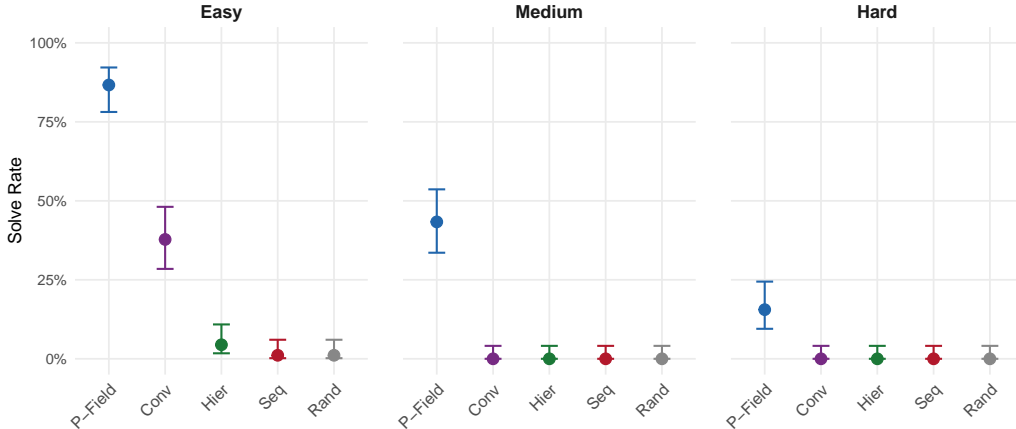

Figure 1: Strategy comparison by difficulty level. Error bars show 95% Wilson Confidence Intervals (CIs). Pressure-field outperforms all baselines at every difficulty level. On medium and hard problems, only pressure-field achieves non-zero solve rates.

6.3 Ablations

6.3.1 Effect of Temporal Decay

Decay proves essential—without it, solve rate drops:

Configuration	Solved/N	Solve Rate	95% CI
Full (with decay)	29/30	96.7%	83.3%–99.4%
Without decay	26/30	86.7%	70.3%–94.7%

Table 4: Decay ablation on easy scheduling problems (30 trials each). Disabling decay reduces solve rate by 10 percentage points.

The effect is meaningful: disabling decay reduces solve rate from 96.7% to 86.7%. Without decay, fitness saturates after initial patches—regions that received early patches retain high fitness indefinitely, making them appear "stable" even when they still contain unscheduled meetings. Since greedy selection prioritizes high-pressure regions, these prematurely-stabilized regions are never reconsidered. This validates the Basin Separation result (Theorem 3): without decay, agents remain trapped in the first stable basin they reach, even if lower-pressure basins exist elsewhere in the landscape.

6.3.2 Effect of Inhibition and Examples

The ablation study tested combinations of decay, inhibition, and few-shot examples on easy scheduling problems:

Feature contributions:

Configuration	Decay	Inhib	Examples	Solve Rate
Full	✓	✓	✓	96.7%
No Decay	✗	✓	✓	86.7%
No Inhibition	✓	✗	✓	96.7%
No Examples	✓	✓	✗	90.0%
Baseline	✗	✗	✗	90.0%

Table 5: Ablation results (30 trials each configuration on easy difficulty).

- **Decay**: +10.0% (full 96.7% vs no_decay 86.7%)
- **Inhibition**: +0.0% (no detectable effect)
- **Examples**: +6.7% (full 96.7% vs no_examples 90.0%)

Decay dominates: configurations with decay achieve higher solve rates. Inhibition shows no detectable effect in this domain, possibly because the 50-tick budget provides sufficient exploration without explicit cooldowns.
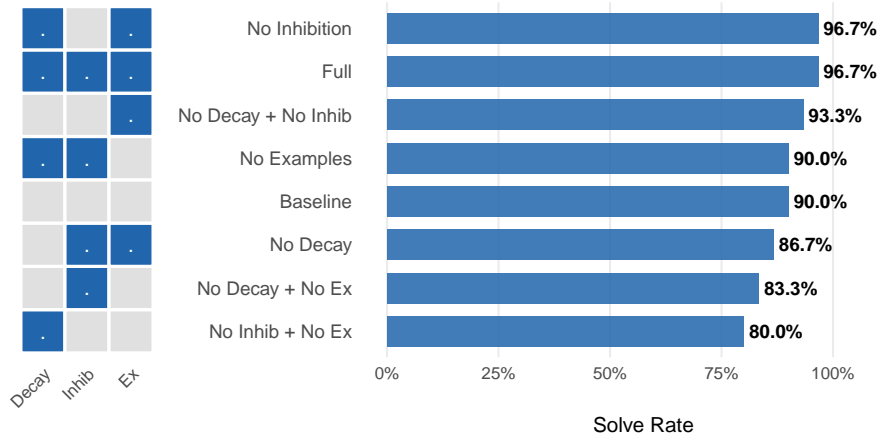

Figure 2: Ablation study results. Left: feature matrix showing which components are enabled. Right: solve rates for each configuration. Decay provides the largest contribution (+10%), followed by examples (+6.7%). Inhibition shows no detectable effect.

6.3.3 Negative Pheromones

In addition to positive pheromones (successful patches stored for few-shot examples), we implement *negative pheromones*: tracking rejected patches that worsened pressure. When agents repeatedly propose ineffective patches (pressure stuck at maximum), the system accumulates rejection history and injects guidance into subsequent prompts.

Unlike the "AVOID" framing that small models (1.5B parameters) struggle to follow, we use *positive language*: rejected empty-room patches become "TIP: Schedule meetings in Room A (improves by X)." This reframes what *not* to do as what *to try instead*.

Negative pheromones decay at the same rate as positive examples (weight $\times 0.95$ per tick, evicted below 0.1), ensuring that old failures don't permanently block valid approaches. Up to 3 recent rejections per region are included in prompts as "Hints for better scheduling."

6.4 Scaling Experiments

Pressure-field maintains consistent performance from 1 to 4 agents on easy difficulty:

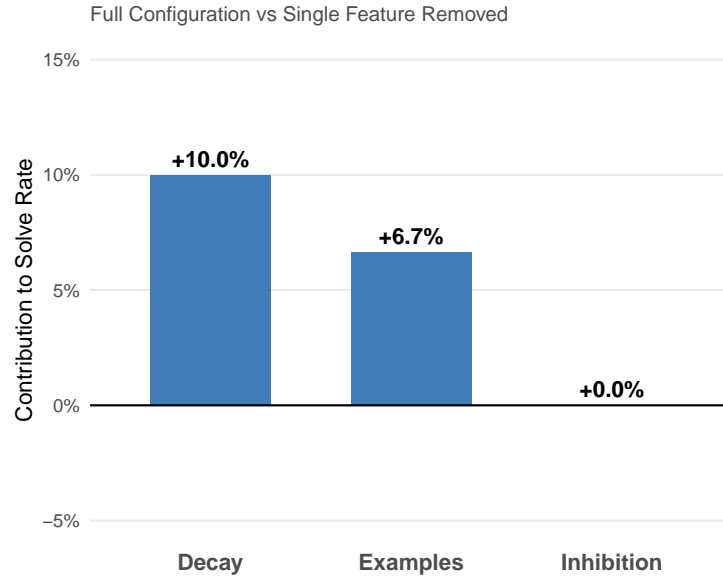The result demonstrates *robustness*: pressure-field coordination maintains consistent solve

Figure 3: Individual feature contributions to solve rate. Decay contributes +10.0%, examples contribute +6.7%, and inhibition shows no measurable effect in this domain.

Agents	Solved/N	Rate	95% CI
1	25/30	83.3%	66.4%–92.7%
2	28/30	93.3%	78.7%–98.2%
4	25/30	83.3%	66.4%–92.7%

Table 6: Pressure-field scaling from 1 to 4 agents (easy difficulty, 30 trials each). Performance remains stable across agent counts.

rates despite 4× variation in agent count. This validates Theorem 4: coordination overhead remains $O(1)$, enabling effective scaling. The slight peak at 2 agents (93.3%) is within Confidence Interval (CI) overlap of 1 and 4 agents, indicating no significant agent-count effect.

6.5 Band and Model Escalation

Ant colonies balance exploitation of known food sources against exploration for new ones. When a pheromone trail grows stale—indicating a depleted source—foragers abandon trail-following and resume random exploration, eventually discovering new paths that become the next generation of trails. This exploitation-exploration balance is fundamental to stigmergic systems: premature commitment to suboptimal solutions must be counteracted by mechanisms that restore exploratory behavior.

Our escalation mechanism implements this principle through two complementary dynamics. *Band escalation* governs the exploitation-exploration trade-off within a single model: when pressure velocity drops to zero (the "trail goes cold"), sampling parameters shift from exploitation (low temperature, focused proposals) through balanced to exploration (high temperature, diverse proposals). This mirrors the ant's behavioral switch from trail-following to random wandering when pheromone signals weaken.

Band	Temperature	Top-p
Exploitation	0.15–0.35	0.80–0.90
Balanced	0.35–0.55	0.85–0.95
Exploration	0.55–0.85	0.90–0.98

Table 7: Sampling parameter ranges per band. Temperature and top-p are randomly sampled within range for diversity. Escalation proceeds Exploitation → Balanced → Exploration as pressure velocity stalls.

Model escalation addresses a different failure mode: when exploration within a model's capability envelope fails to discover productive paths, the system recruits more capable agents. This mirrors the immune system's escalation from innate to adaptive response: macrophages and neutrophils handle routine pathogens quickly and cheaply, but when these first responders fail to clear an infection, the system escalates to T-cells and antibody-producing B-cells—slower to mobilize but capable of addressing threats invisible to simpler mechanisms. Similarly, model escalation (0.5b → 1.5b → 3b) reserves greater reasoning capacity for regions that resist simpler approaches. Each model upgrade resets to exploitation band, giving the more capable model opportunity to exploit solutions invisible to its predecessor before resorting to exploration.

This two-level mechanism—behavioral adaptation within agents (band escalation) and capability escalation across agents (model escalation)—maintains the stigmergic principle: coordination emerges from environment signals (pressure gradients) rather than explicit planning. The system does not "decide" to explore or escalate; it reacts to pressure stagnation, just as ants react to pheromone decay.

6.6 Difficulty Scaling

Performance varies across difficulty levels, revealing the unique strength of pressure-field coordination:

The difficulty scaling reveals critical insights:

1. **Pressure-field is the only strategy that scales**: While all strategies degrade on harder problems, pressure-field maintains meaningful solve rates (43.3% medium, 15.6% hard) where all baselines achieve 0%.

2. **The gap widens with difficulty**: On easy problems, pressure-field leads by 48.9

Difficulty	Pressure-field	Conversation	Hierarchical	Sequential/Random
Easy	86.7% (78/90)	37.8% (34/90)	4.4% (4/90)	1.1% (1/90)
Medium	43.3% (39/90)	0.0% (0/90)	0.0% (0/90)	0.0% (0/90)
Hard	15.6% (14/90)	0.0% (0/90)	0.0% (0/90)	0.0% (0/90)
Total	**48.5%** (131/270)	**12.6%** (34/270)	**1.5%** (4/270)	**0.4%** (1/270)

Table 8: Solve rate by difficulty level (90 trials each per difficulty, 270 per strategy total). Only pressure-field solves medium and hard problems. See Table 3 for confidence intervals.

percentage points over conversation (86.7% vs 37.8%). On medium and hard problems, the gap becomes absolute—pressure-field solves problems that no baseline can solve.

3. **Effect sizes are large**: Cohen's h on easy problems: $h = 1.07$ vs conversation, $h = 1.97$ vs hierarchical, $h = 2.18$ vs sequential/random.
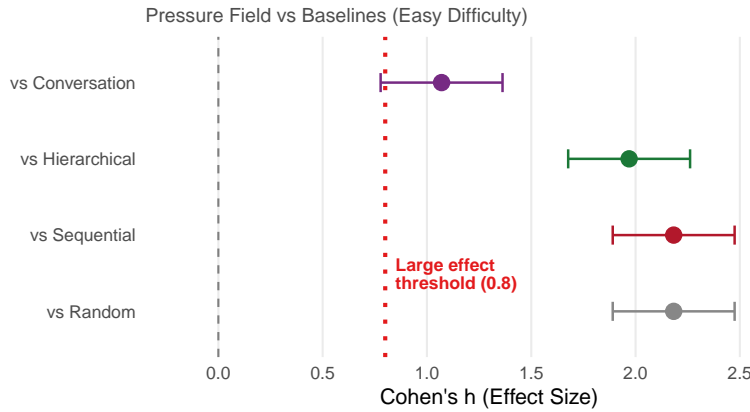

Figure 4: Effect sizes (Cohen's h) for pressure-field versus each baseline on easy problems. The dashed line indicates the "large effect" threshold ($h = 0.8$). All comparisons exceed this threshold, with effects ranging from $h = 1.07$ (vs conversation) to $h = 2.18$ (vs sequential/random).

6.7 Convergence Speed

For solved cases, pressure-field converges faster than baselines on easy problems and maintains consistent convergence speed across difficulty levels:

Strategy	Easy	Medium	Hard
Pressure-field	17.8 (n=78)	34.6 (n=39)	32.3 (n=14)
Conversation	29.9 (n=34)	—	—
Hierarchical	40.0 (n=4)	—	—

Table 9: Average ticks to solution by difficulty (solved cases only). Only pressure-field solves medium and hard problems. Dashes indicate no solved cases.

On easy problems, pressure-field solves 1.7× faster than conversation and 2.2× faster than hierarchical. Notably, pressure-field's convergence speed on hard problems (32.3 ticks) is comparable to medium problems (34.6 ticks)—the hard problems that *do* get solved converge at similar rates, suggesting that solvability rather than convergence speed is the limiting factor on difficult problems. This bimodal pattern—fast convergence when solvable, complete failure otherwise—suggests that model capability rather than search time is the limiting factor on hard problems.
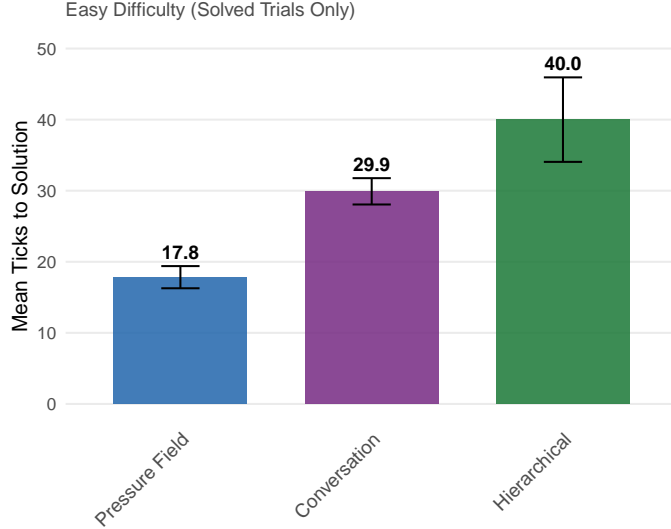
Figure 5: Mean ticks to solution on easy problems (solved cases only). Error bars show standard error. Pressure-field converges fastest (17.8 ticks), followed by conversation (29.9 ticks) and hierarchical (40.0 ticks).

6.8 Final Pressure Analysis

For both solved and unsolved cases, final pressure reveals solution quality:

Strategy	Easy	Medium	Hard
Pressure-field	0.13	1.02	2.48
Conversation	26.3	58.2	90.1
Hierarchical	28.6	59.4	88.0

Table 10: Average final pressure by difficulty (lower is better).

Pressure-field achieves $200\times$ lower final pressure on easy, $57\times$ lower on medium, and $35\times$ lower on hard problems compared to conversation and hierarchical baselines. Even when pressure-field does not fully solve a problem, it achieves much better partial solutions.

7 Discussion

7.1 Why Does Pressure-Field Dominate?

Our results contradict the intuition that explicit coordination should outperform implicit coordination. We identify three factors explaining pressure-field's dominance:

Coordination overhead harms performance. Hierarchical systems spend computational budget on coordination rather than problem-solving. The manager-worker protocol requires multiple LLM calls per patch (planning, delegation, execution), while pressure-field requires only one (patch proposal). This overhead compounds: hierarchical attempts fewer patches per tick, reducing exploration.

Local greedy decisions are effective for constraint satisfaction. Meeting room scheduling exhibits locality: fixing a conflict in one time block rarely creates conflicts in distant blocks. This matches pressure-field's locality assumption, making greedy local optimization effective. Hierarchical coordination's global planning provides no benefit for locally-decomposable problems.

Parallel validation amplifies pressure-field's advantage. Pressure-field validates patches for multiple regions simultaneously, applying the highest-scoring patch per region that reduces pressure. Hierarchical validates one patch at a time, requiring multiple ticks to explore alternatives. On problems with many valid solutions, parallel exploration finds solutions faster.

7.2 Failure Analysis: Why Hierarchical Collapses

The 30× performance gap (48.5% vs 1.5%) demands deeper investigation. Analysis of hierarchical trials reveals a catastrophic failure pattern: the *rejection loop.*

The rejection loop mechanism. Hierarchical always selects the highest-pressure region for improvement. But the highest-pressure regions are high-pressure precisely because they are difficult to improve. When the LLM proposes a patch that fails validation (does not reduce pressure), the region remains highest-pressure and is selected again next tick. This creates a self-reinforcing cycle: 66.7% of hierarchical runs (180/270) applied zero patches across all 50 ticks, stuck targeting the same intractable region repeatedly.

Rejection rates confirm the pattern. Across all hierarchical trials, only 173 patches were accepted out of 13,460 proposed—a 98.7% rejection rate. By contrast, pressure-field's parallel exploration means that even if one agent's patch is rejected, other agents make progress on different regions. The non-blocking architecture prevents any single difficult region from stalling the entire system.

The architectural lesson. Hierarchical's design embodies a reasonable intuition: focus intelligent effort on the worst problems. But this creates a trap when combined with strict validation: the hardest problems resist improvement, causing repeated rejection, which blocks progress everywhere. Pressure-field avoids this trap through distributed exploration— progress happens where it can, not where a central planner dictates it must.

7.3 Limitations

Our experiments reveal several important limitations:

Absolute solve rates are modest on hard problems. Even pressure-field achieves only 15.6% on hard problems. Meeting room scheduling with tight constraints (5 rooms, 60 meetings, 30% pre-scheduled) remains challenging for small models (0.5b–3b parameters); larger models may achieve higher absolute solve rates.

Domain specificity. Results on meeting room scheduling may not generalize to domains lacking measurable pressure gradients or locality properties. Tasks requiring global planning or long-horizon reasoning may favor hierarchical approaches.

Additional practical limitations:

- Requires well-designed pressure functions (not learned from data)
- Decay rates $\lambda_f, \lambda_\gamma$ and inhibition period require task-specific tuning
- May not suit tasks requiring long-horizon global planning
- Goodhart's Law: agents may game poorly-designed metrics
- Resource cost of parallel validation: testing K patches requires $O(K \cdot |A|)$ memory where $|A|$ is artifact size

7.4 When to Choose Each Approach

Our results suggest the following guidance:

Pressure-field coordination is preferable when:

1. **Performance matters.** Pressure-field achieves 3–30× higher solve rates than alternatives.

2. **Simplicity is valued.** No coordinator agent needed; coordination emerges from shared state.

3. **Fault tolerance matters.** No single point of failure; agents can join/leave without protocol overhead.

4. **Pressure signals are available.** The domain provides measurable quality gradients.

5. **Problems are locally decomposable.** Local fixes improve global quality without cascading conflicts.

Hierarchical coordination may be appropriate when:

1. **Explicit control is required.** Some domains require deterministic task assignment for regulatory or safety reasons.

2. **Interpretability is critical.** Hierarchical task assignment provides clear audit trails.

3. **Global planning is essential.** Tasks with strong non-local dependencies may benefit from centralized reasoning.

7.5 Band and Model Escalation as Adaptive Capability

All experiments use a two-level escalation mechanism. *Band escalation* cycles through sampling strategies (Exploitation → Balanced → Exploration, 7 ticks each) before *model escalation* progresses through model sizes (0.5b → 1.5b → 3b parameters). Model escalation triggers when regions remain high-pressure for 21 consecutive ticks.

This mechanism proves beneficial for pressure-field: on hard problems, pressure-field achieves 15.6% with escalation enabled. The escalation mechanism works because larger models have broader solution coverage and different sampling bands explore different regions of solution space.

7.6 Future Work

- **Learned pressure functions**: Current sensors are hand-designed. Can we learn pressure functions from solution traces?

- **Adversarial robustness**: Can malicious agents exploit pressure gradients to degrade system performance?

- **Multi-artifact coordination**: Extension to coupled artifacts where patches in one affect pressure in another

- **Larger-scale experiments**: Testing on schedules with more rooms and longer time horizons to characterize scaling limits

- **Alternative domains**: Applying pressure-field coordination to code refactoring, configuration management, and other artifact refinement tasks

7.7 Societal Implications

Pressure-field coordination raises societal concerns that extend beyond technical performance. We identify three critical issues—accountability attribution, metric gaming through Goodhart's Law, and explainability challenges—that require deliberate design choices in deployment.

7.7.1 Accountability and Attribution

When coordination emerges from shared pressure gradients rather than explicit delegation, attributing outcomes to individual agents becomes challenging. In hierarchical systems, task assignment creates clear accountability chains. In pressure-field coordination, multiple agents may contribute to a region through independent pressure-reducing actions, with no record of which agent "owned" the outcome.

This accountability diffusion has both benefits and risks. The benefit is fault tolerance:

agent failures degrade performance gracefully rather than catastrophically. The risk is opacity in failure analysis: identifying which agent proposed a problematic patch—and what pressure signal motivated it—requires detailed logging that the minimal coordination mechanism does not inherently provide.

For deployment in regulated domains, this suggests an augmentation requirement: pressure-field systems must maintain audit logs recording patch provenance, pressure signals at proposal time, and validation outcomes. The coordination mechanism remains simple—agents coordinate through shared state—but operational deployment adds logging infrastructure preserving accountability.

7.7.2 Goodhart's Law and Metric Gaming

Goodhart's Law states: "When a measure becomes a target, it ceases to be a good measure." Pressure-field coordination is vulnerable to this dynamic because agents are optimized to reduce pressure as defined by designer-specified functions. If those functions imperfectly capture true quality—and they inevitably do—agents will discover and exploit the mismatch.

Consider code quality pressure functions penalizing complexity metrics. An agent might reduce complexity by splitting functions excessively, harming readability while improving the metric. The mitigation is not abandoning pressure functions but designing them defensively: use multiple orthogonal pressure axes, include adversarial sensors detecting gaming strategies, and audit whether pressure reduction correlates with human quality judgments. Pressure functions should evolve as agents discover exploits.

Foundation models introduce second-order gaming concerns: LLMs trained on internet-scale text may have implicit knowledge of how to game specific benchmarks. This suggests pressure functions for LLM-based systems should favor domain-specific quality signals harder to optimize without genuine improvement.

7.7.3 Explainability Challenges

In hierarchical systems, explanations follow delegation chains: "Manager X assigned task Y to Worker Z because condition C held." In pressure-field coordination, the explanation is: "Region R had high pressure, agent A proposed patch Δ reducing pressure by δ." This is mechanistically transparent but causally opaque—it describes what happened without explaining why that particular patch was chosen.

This is the explainability trade-off inherent to emergent coordination: simplicity in mechanism comes at the cost of legibility in rationale. For many domains—code formatting, resource optimization, routine maintenance—the trade-off is acceptable: outcomes are verifiable even if reasoning is opaque. For high-stakes domains requiring human oversight, opacity is unacceptable.

The design implication is domain-dependent deployment: pressure-field coordination suits domains where outcome verification is cheap even if reasoning transparency is limited. For domains requiring justification to human stakeholders, hierarchical coordination remains necessary despite overhead costs.

7.7.4 Design Implications

These concerns suggest three requirements for responsible deployment: comprehensive audit logging preserving patch provenance and pressure signals, defensive pressure function design with multiple orthogonal axes, and domain-appropriate verification matching coordination opacity with outcome verifiability. The coordination mechanism remains simple—but responsible deployment requires surrounding infrastructure addressing accountability, gaming, and explainability.

8 Conclusion

We presented pressure-field coordination, a decentralized approach to multi-agent systems that achieves coordination through shared state and local pressure gradients rather than explicit orchestration.

Our theoretical analysis establishes convergence guarantees under pressure alignment conditions, with coordination overhead independent of agent count. Empirically, on meeting room scheduling across 1350 trials, we find:

1. **Pressure-field outperforms all baselines** (48.5% vs 12.6% for conversation, 1.5% for hierarchical, all $p < 0.001$). Implicit coordination through shared pressure gradients exceeds explicit hierarchical coordination.

2. **Pressure-field is the only strategy that scales to harder problems**. On medium and hard problems, pressure-field achieves 43.3% and 15.6% solve rates respectively, while all baselines achieve 0%.

3. **Temporal decay is essential**. Disabling it reduces solve rate by 10 percentage points, trapping agents in local minima.

This work demonstrates that implicit coordination can outperform explicit coordination for constraint satisfaction tasks. Pressure-field achieves this with simpler architecture: no coordinator agent, no explicit message passing, just shared state and local pressure gradients.

FMs' zero-shot capabilities eliminate the need for domain-specific action representations; pressure-field coordination eliminates the need for complex multi-agent protocols; together they enable simple multi-agent systems.

These results suggest that for domains with measurable quality signals and locally-decomposable structure, implicit coordination through shared state offers not just a simpler but a more effective alternative to explicit hierarchical control.

A Experimental Protocol

This appendix provides complete reproducibility information for all experiments.

A.1 Hardware and Software

Hardware: NVIDIA RTX 4070 8GB GPU, AMD Ryzen 9 7940HS, 64GB RAM

Software:

- Rust 1.75+ (edition 2024)
- Ollama (local LLM inference server)
- Models: `qwen2.5:0.5b`, `qwen2.5:1.5b`, `qwen2.5:3b`

A.2 Model Configuration

Models are served via Ollama with a system prompt configured for schedule optimization:

```
You optimize meeting room schedules. Given a schedule with gaps or conflicts,
propose ONE change: move, swap, or reschedule a meeting to reduce gaps,
overlaps, and utilization variance. Return ONLY your proposed change
in the format: MOVE meeting_id TO room day start_time
```

For multi-model setups (model escalation), models share a single Ollama instance with automatic routing based on model name.

A.3 Sampling Diversity

The experiment framework overrides default sampling parameters with three exploration bands per LLM call:

Band	Temperature	Top-p
Exploitation	0.15–0.35	0.80–0.90
Balanced	0.35–0.55	0.85–0.95
Exploration	0.55–0.85	0.90–0.98

Table 11: Sampling parameter ranges. Each LLM call randomly samples from one band.

This diversity prevents convergence to local optima and enables exploration of the solution space.

A.4   Baseline Implementation Details

To ensure fair comparison, all strategies use identical LLM prompts and parsing logic. The only differences are region selection and patch validation:

Prompt (identical across all strategies):

```
You are a meeting room scheduler. Output schedules in the exact format requested.

Meeting Room Schedule Optimization.
Goal: Schedule meetings to minimize gaps and avoid conflicts.

Time Block: [time range]

Rooms:
  Room A: capacity 10
  Room B: capacity 8
  ...

Current assignments:
[current schedule for this time block]

Unscheduled meetings that could fit in this block:
  Meeting 5: 60min, 4 attendees
  Meeting 12: 30min, 2 attendees
  ...

Constraints:
- No attendee can be in multiple meetings at the same time
- Room capacity must fit attendees

Output the schedule for this time block.
```

Region selection per strategy:

- **Pressure-field**: Multiple agents select regions in parallel based on pressure gradients
- **Hierarchical**: Always selects the single highest-pressure region
- **Sequential**: Round-robin through regions in fixed order
- **Random**: Uniform random region selection

Patch validation per strategy:

- **Pressure-field**: Validates all patches; applies only those reducing pressure
- **Hierarchical**: Validates single patch; applies only if pressure reduces
- **Sequential/Random**: No validation; applies any syntactically valid patch

Why "one patch per tick" for hierarchical? This asymmetry is inherent to the

coordination paradigm, not an implementation choice. Hierarchical control requires the manager to: (1) observe global state, (2) select a target region, (3) delegate to a worker, (4) receive the proposed patch, and (5) validate before applying. Delegating to K workers simultaneously would require additional coordination protocols—work distribution to avoid conflicts, result aggregation, tie-breaking when multiple patches target overlapping state— that would transform hierarchical control into a fundamentally different architecture. The sequential bottleneck is the cost of centralized decision-making. Pressure-field avoids this overhead through stigmergic coordination: agents observe shared state independently and propose patches without central delegation, enabling natural parallelism.

Conversation baseline implementation The conversation baseline implements AutoGen-style multi-agent dialogue with three roles: a *Coordinator* that selects which region to address, a *Proposer* that suggests schedule changes, and a *Validator* that critiques proposals. Each tick proceeds as follows: (1) the Coordinator identifies a high-pressure region (1 LLM call), (2) the Proposer and Validator engage in up to 5 dialogue turns (2 calls per turn), (3) if the Validator approves, the patch is applied. This yields 4–12 LLM calls per tick for a single region, compared to pressure-field's K parallel calls across K regions. The conversation terminates when the Validator approves or the turn limit is reached. We use fixed sampling parameters (T=0.3, top-p=0.9) for structured dialogue, lower than pressure-field's exploration bands.

A.5 Problem Generation and Seeding

Fair strategy comparison requires identical problem instances: each strategy must face the same scheduling challenge within a trial. We achieve this through deterministic seeding.

Each trial generates its problem from a seed:

$$\text{seed} = \text{trial} \times 1000 + \text{agent\_count}$$

Trial 5 with 2 agents yields seed 5002, producing identical meeting configurations whether evaluated with pressure-field, conversation, or hierarchical coordination.

The seed governs all stochastic generation:

- Meeting durations (1–4 time slots)
- Attendee assignments (2–5 participants)
- Room preferences and capacity requirements
- Pre-scheduled vs. unassigned meeting distribution
- Time slot availability patterns

A.6 Experiment Commands

Main Grid (Strategy Comparison):

```
schedule-experiment --host http://localhost:11434 \
  grid --trials 30 \
  --strategies pressure_field,sequential,random,hierarchical,conversation \
  --agents 1,2,4 --difficulties easy,medium,hard \
  --max-ticks 50
```

Ablation Study:

```
schedule-experiment --host http://localhost:11434 \
  ablation --trials 30 --agents 2 --difficulty easy --max-ticks 50
```

Scaling Analysis:

```
schedule-experiment --host http://localhost:11434 \
  grid --trials 30 \
  --strategies pressure_field \
  --agents 1,2,4 --difficulties easy \
  --max-ticks 50
```

Band and Model Escalation:

```
# Full escalation chain enabled by default
# Band escalation: Exploitation -> Balanced -> Exploration (7 ticks each)
# Model escalation: 0.5b -> 1.5b -> 3b (after 21 ticks at high pressure)
schedule-experiment --host http://localhost:11434 \
  grid --trials 30 --agents 2 --difficulties medium \
  --max-ticks 100
```

Difficulty Scaling:

```
# Easy: 3 rooms, 20 meetings, 70% pre-scheduled
schedule-experiment --host http://localhost:11434 \
  grid --trials 30 --agents 2 --difficulties easy --max-ticks 50

# Medium: 5 rooms, 40 meetings, 50% pre-scheduled
schedule-experiment --host http://localhost:11434 \
  grid --trials 30 --agents 2 --difficulties medium --max-ticks 50

# Hard: 5 rooms, 60 meetings, 30% pre-scheduled
schedule-experiment --host http://localhost:11434 \
  grid --trials 30 --agents 2 --difficulties hard --max-ticks 100
```

A.7 Metrics Collected

Each experiment records:

- `solved`: Boolean indicating all meetings scheduled with zero overlaps
- `total_ticks`: Iterations to solve (or max if unsolved)
- `pressure_history`: Pressure value at each tick (gaps + overlaps + util_var + unscheduled)
- `band_escalation_events`: Sampling band changes (tick, from_band, to_band)
- `model_escalation_events`: Model tier changes (tick, from_model, to_model)
- `final_model`: Which model tier solved the schedule
- `token_usage`: Prompt and completion tokens consumed (pressure-field only; conversation baseline uses a different LLM client path that does not collect token statistics, precluding cross-strategy comparison)

A.8 Replication Notes

Each configuration runs 30 independent trials with different random seeds to ensure reliability. Results report mean solve rates and tick counts across trials.

A.9 Estimated Runtime

References

[CrewAI(2024)] CrewAI. 2024. Framework for orchestrating role-playing, autonomous AI agents. https://github.com/crewAIInc/crewAI.

Experiment	Configurations	Trials	Est. Time
Main Grid	45	30	5 hours
Ablation	5	30	1 hour
Scaling	3	30	45 min
Difficulty	15	30	2.5 hours
Total			**~9 hours**

Table 12: Estimated runtime for all experiments on NVIDIA RTX 4070 8GB GPU.

[De Wolf and Holvoet(2005)] Tom De Wolf and Tom Holvoet. 2005. Towards a Methodology for Engineering Self-Organising Emergent Systems. In *Self-Organization and Autonomic Informatics (I)*. IOS Press.

[Decker and Lesser(1995)] Keith Decker and Victor Lesser. 1995. Designing a Family of Coordination Algorithms. In *Proceedings of the First International Conference on Multi-Agent Systems (ICMAS-95)*. 73–80.

[Dorigo and Gambardella(1997)] Marco Dorigo and Luca Maria Gambardella. 1997. Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation* 1, 1 (1997), 53–66.

[Dorigo et al.(1996)] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. 1996. Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B* 26, 1 (1996), 29–41.

[Grassé(1959)] Pierre-Paul Grassé. 1959. La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp. La théorie de la stigmergie. *Insectes Sociaux* 6 (1959), 41–80.

[Grosz and Kraus(1996)] Barbara J. Grosz and Sarit Kraus. 1996. Collaborative Plans for Complex Group Action. *Artificial Intelligence* 86, 2 (1996), 269–357.

[Hong et al.(2023)] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. 2023. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework. *arXiv preprint arXiv:2308.00352* (2023).

[Horling and Lesser(2004)] Bryan Horling and Victor Lesser. 2004. A Survey of Multi-agent Organizational Paradigms. *The Knowledge Engineering Review* 19, 4 (2004), 281–316.

[Li et al.(2023)] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. 2023. CAMEL: Communicative Agents for "Mind" Exploration of Large Language Model Society. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[Malone and Crowston(1994)] Thomas W. Malone and Kevin Crowston. 1994. The Interdisciplinary Study of Coordination. *Comput. Surveys* 26, 1 (1994), 87–119.

[Monderer and Shapley(1996)] Dov Monderer and Lloyd S. Shapley. 1996. Potential Games. *Games and Economic Behavior* 14 (1996), 124–143.

[Nedić and Ozdaglar(2009)] Angelia Nedić and Asuman Ozdaglar. 2009. Distributed subgradient methods for multi-agent optimization. *IEEE Trans. Automat. Control* 54, 1 (2009), 48–61.

[Shoham and Leyton-Brown(2008)] Yoav Shoham and Kevin Leyton-Brown. 2008. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press.

[Wu et al.(2023)] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. *arXiv preprint arXiv:2308.08155* (2023).

[Yuan et al.(2016)] Kun Yuan, Qing Ling, and Wotao Yin. 2016. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization* 26, 3 (2016), 1835–1854.