Optimization-Based Coordination of Large-Scale Multi-Agent AI Systems: A Technical Survey and Research Roadmap

Executive Summary

This report provides a comprehensive technical survey and research gap analysis for the optimization-based coordination of the ATLAS/TURING multi-agent AI research system. The coordination of 40+ specialized agents presents a multi-faceted optimization challenge, spanning static assignment, dynamic scheduling, resource allocation, architecture discovery, and system robustness.

The analysis indicates that the field is at a critical inflection point, moving from manually-designed, static coordination workflows (e.g., simple agent chains) to *dynamically optimized, self-evolving, and robust agentic architectures*. The state-of-the-art (SOTA) in 2024-2025 is defined by (1) the application of classical Operations Research (OR) techniques (e.g., Quadratic Assignment, Bandits-with-Knapsacks) in novel, *contextual*, and *learned* settings; (2) the emergence of powerful evolutionary frameworks (EvoFlow, EvoAgentX) and Neural Architecture Search (NAS) analogues (MASS) that *auto-discover* workflow topologies; and (3) a new, urgent focus on robustness, addressed via adversarial testing (DeepTeam, AgentSafety) and min-max optimization (RobustFlow).

The primary recommendation is to architect the ATLAS/TURING system as a *hybrid* of these approaches:

1. **For Task Assignment:** Utilize **Market-Based Mechanisms (Combinatorial Auctions)** for decentralized, scalable assignment, falling back on **ML-hybridized QAP metaheuristics** (e.g., learned cost functions) for centralized, synergy-critical subproblems.
2. **For Workflow Routing:** Employ a **stateful, cyclical graph framework** (i.e., **LangGraph**) to manage the core "dialectical chains" (Designer$\rightarrow$Critic$\rightarrow$Refactorer). This loop should be formally modeled as a **Stochastic Shortest Path problem**, with Reinforcement Learning (RL)

used to learn the optimal "stop/continue" policy.
3. **For Resource Allocation:** Implement **Hierarchical Contextual Bandits with Knapsacks (HBwK)**, using Thompson Sampling to manage exploration/exploitation at both the *agent-group* and *individual-agent* levels.
4. **For Performance Evolution:** Use a **Population-Based Training (PBT)** scheduler (from Ray Tune) to continuously optimize agent hyperparameters (prompts, model configs), using an **Elo rating system** (fed by Validator scores) as the fitness function.
5. **For Architecture Optimization:** Build a custom auto-discovery layer based on SOTA frameworks like **EvoFlow** and **MASS** to *evolve* the coordination graphs themselves, using *constrained evolutionary operators* to ensure system safety and validity.
6. **For Robustness:** Establish a "Red Team" subsystem using frameworks like **DeepTeam** to continuously generate adversarial tests. These tests will serve as the *inner loop* of a **bi-level min-max optimization** process to proactively design failure-resistant workflows.

This report is structured in three parts. Part 1 is a technical survey of the methods. Part 2 analyzes the tools and frameworks for implementation. Part 3 details seven novel, high-impact research contributions, providing a concrete publication roadmap.

Part 1: A Technical Survey of Multi-Agent Coordination and Optimization

Section 1: Agent-Task Assignment Optimization

The problem of assigning N agents to M tasks is a foundational challenge in combinatorial optimization (CO). In its most sophisticated form, the Quadratic Assignment Problem (QAP), the objective is to minimize a cost function that accounts for both the linear cost of assigning an agent to a task and the quadratic, interdependent costs that arise from synergies or conflicts between pairs of agents assigned to pairs of tasks.

1.1 Classical QAP: Exact, Metaheuristic, and Learned Approaches

The QAP was formally introduced in 1957 by Koopmans and Beckmann in the context of

facility location.[1] It is NP-hard and widely regarded as one of the most difficult CO problems to solve in practice.[2] Given n facilities (agents) and n locations (tasks), along with two $n \times n$ matrices F (flow, e.g., required information transfer between agents) and D (distance, e.g., communication cost or dissimilarity between tasks), the objective is to find a permutation ϕ of $\{1,..., n\}$ that minimizes the total cost [2]:

$$\min_{\phi \in \Phi_n} \sum_{i=1}^{n} \sum_{j=1}^{n} F_{i,j} D_{\phi(i),\phi(j)}$$

Exact Methods:

Exact algorithms, such as branch-and-bound and cutting-plane methods, aim to find the provably optimal solution. However, their computational complexity makes them unusable for large-scale problems. The largest, non-trivial instances from the benchmark QAPLIB (a standard collection of QAP instances 4) that have been solved to optimality are only around $n=36$.2 For a system of $N=40+$ agents, these methods are computationally infeasible for real-time (or even offline) assignment.

Metaheuristics:

For practical purposes, heuristic and metaheuristic algorithms are the standard approach for large-scale QAPs. They sacrifice provable optimality for the ability to find high-quality solutions in a feasible amount of time.3

- **Algorithms:** The most widely applied metaheuristics for QAP include Tabu Search (TS), Simulated Annealing (SA), and Genetic Algorithms (GA).[3] Comparative studies often find that TS, in particular, demonstrates very good performance across a range of QAP instances.[6]
- **Implementations:** The QAPSolver C++ library provides accessible implementations of these heuristics, allowing for rapid prototyping (e.g., model.TS(initial, 50, 7, 10)).[4]
- **Performance:** The performance bottleneck for large instances remains significant. 2021 research highlights the necessity of using massively parallel implementations (e.g., on GPUs) to reduce execution time.[5] This performance drive continues, with 2025 research exploring novel local search heuristics specifically designed for in-memory computing (IMC) hardware, which can compute full neighborhoods simultaneously to accelerate the search.[1]

State-of-the-Art Solvers (2024):

A search for "QAP solvers" reveals a critical distinction. The most active and well-benchmarked Python ecosystem is qpsolvers 7 and its associated benchmark suite, qpbenchmark.8 This ecosystem provides a unified Python wrapper for a wide array of high-performance QP solvers, including Clarabel, OSQP, PIQP, ProxQP, Gurobi, and MOSEK.8 However, these are solvers for *general* Quadratic Programming (QP), which includes a broad class of problems. QAP is a specific, highly-structured *integer* QP with permutation constraints. General-purpose QP solvers are not typically optimized for this specific structure and are often intended for continuous or less-constrained convex problems.

The true SOTA for large-scale QAP falls into two categories:

1. **Commercial Solvers:** Commercial Mixed-Integer Programming (MIP) solvers like Gurobi and CPLEX have exceptionally powerful branch-and-cut algorithms that can be applied to the QAP (typically by linearizing the quadratic objective).[10]
2. **Specialized Metaheuristics:** As discussed, highly-tuned parallel implementations of TS, SA, or GAs.[5]
3. **Machine Learning for CO (ML4CO):** A new SOTA direction involves using deep learning, particularly Graph Neural Networks (GNNs), to *learn* heuristics that solve CO problems. The awesome-ml4co repository lists several key papers (ICML 2023, AAAI 2024) where models learn to solve routing and assignment problems.[11] This hybrid approach is a highly promising avenue for the ATLAS system.

1.2 Extensions for Dynamic and Contextual AI Contexts

The primary limitation of classical QAP is its static nature. The F and D matrices are assumed to be fixed and known in advance.[2] In the ATLAS system, this assumption is invalid. The "flow" (synergy) between a Designer and a Critic and the "cost" (e.g., predicted time) of assigning the Critic to a task are *dynamic*. They depend on:

- **System State:** Current load, queue length, agent availability.
- **Agent Confidence:** The Designer's confidence in its last artifact.
- **Historical Performance:** The track record of this specific agent pairing.

This defines a **Contextual QAP**, a significant research gap. While the 2024 literature is replete with "context-aware" multi-agent systems [12] and dynamic agent orchestration [15], the formal optimization problem of "Online Contextual QAP" is not well-established.

The SOTA literature (2024-2025) suggests that for highly dynamic LLM-based agent systems, the assignment problem is not being solved as a single, large-scale, global QAP. Instead, it is being *reframed* as a sequential, dynamic routing or task decomposition problem.

- The **TDAG framework** (2024) focuses on *dynamic* task decomposition and sub-agent generation, emphasizing "adaptability and context awareness".[16] This implies the set of tasks M is itself not static.
- Recent agent architectures (2024) feature components like a "Delegator" that "is responsible for assigning tasks to the appropriate agents or tools" based on the current context and graph state.[18]
- This approach transforms the single, intractable global QAP into a series of smaller, local, and context-dependent assignment decisions, which are more tractable and adaptive.

1.3 Auction and Market Mechanisms

As a scalable, decentralized alternative to a centralized QAP solver, Auction-Based Task Allocation (ATA) is a highly effective mechanism.[19] In this model, the 40+ agents act as self-interested actors that *bid* for tasks.

- **Combinatorial Auctions:** This is the most relevant formulation for the ATLAS system. It allows agents to place bids on *bundles* of tasks.[21] For example, an agent might bid: "I will do Literature_Review (Task 1) and Hypothesis_Generation (Task 2) for a combined cost of X, which is less than the sum of their individual costs." This bidding language directly captures the *synergies* that the QAP models via its quadratic term.
- **State-of-the-Art (2024):** This is a very active field.
    - A 2023 paper proposes an architecture that *unifies* market-inspired combinatorial auctions with Behavior Trees for task execution, specifically designed to handle multi-stage, dynamic tasks.[21]
    - A 2024 survey paper from IJCAI [22] explores "Intelligent Agents for Auction-based Federated Learning (IA-AFL)." This serves as a strong analogue, where "data owners" (agents) bid to participate in "FL tasks" (research tasks). This work highlights the role of intelligent agents in supporting the stakeholders (bidders, sellers, auctioneer) in their decision-making.[22]
    - ATA is also heavily used in multi-robot and UAV swarm coordination (2024), where distributed, auction-based algorithms are essential for dynamic environments with limited communication.[19]

For a system with 40+ agents, a centralized QAP optimizer represents a single point of failure and a severe scalability bottleneck. An auction mechanism is far more robust and scalable. Furthermore, it addresses the *incentive problem*. By designing an incentive-compatible mechanism (e.g., a Vickrey-Clarke-Groves auction), the system can encourage agents to report their *true* costs and capabilities, which is a major challenge for a central QAP solver that must *estimate* all costs.

1.4 Multi-Objective Assignment

The ATLAS system must optimize for multiple, conflicting objectives: maximizing research quality, minimizing completion time, and minimizing resource cost (e.g., GPU usage, API credits).[23]

- **Pareto Optimality:** The goal is not a single "best" solution, but the set of all *Pareto-optimal* solutions—the **Pareto frontier**. A solution is Pareto-optimal if no single

objective can be improved without degrading at least one other objective.[23]

- **Methods:**
  1. **Multi-Objective Reinforcement Learning (MORL):** This is the SOTA *learning* approach. MORL algorithms are designed to learn a *set* of optimal policies (a "Pareto set") simultaneously, rather than a single one.[25] This allows the system to dynamically adapt to different user preferences (e.g., "fastest" vs. "highest quality") without retraining.[25] However, MORL can suffer from low efficiency in finding *dense* Pareto solutions.[26]
  2. **Multi-Objective Evolutionary Algorithms (MOEAs):** Evolutionary methods like NSGA-II are highly effective at discovering Pareto fronts in complex, multi-modal search spaces.[27] This approach is central to frameworks like EvoFlow (see Section 4.1).
  3. **Weighted Sum:** The simplest method, where objectives are scalarized with weights (e.g., TotalCost = w_1 * speed + w_2 * (1-quality) + w_3 * compute). This is a common technique used in reinforcement learning reward functions.[23]
- **Constraint Handling:** Hard constraints (e.g., "The Validator agent *must* be included in the workflow") are handled differently. In classical OR solvers (CPLEX, Gurobi, OR-Tools) [10], these are added as linear constraints to the model. In MOEAs or MORL, they are handled via constrained optimization techniques, such as penalty functions or "safe" operators that ensure solutions remain in the feasible space.

Table 1.1: Comparison of Agent-Task Assignment Approaches

Method	Scalability (Max N)	Optimality	Dynamics	Architecture	SOTA Implementation (2024/2025)	Key Limitation
Exact QAP Solver	Low (N $\approx$ 36) [2]	Exact	Static	Centralized	Gurobi, CPLEX [10]	Intractable for 40+ agents.
QAP Metaheuristic	Medium-High	Approximate	Static	Centralized	QAPSolver (C++) [4], OR-Tools [28]	Assumes static, known cost

						matrices.
ML-Learned Heuristic	High	Approximate	Static	Centralized	ML4CO [11]	Learns to solve a *static* problem; doesn't handle dynamic contexts.
Contextual MAB	Very High	Approximate	Dynamic	Decentralized	LinUCB [29]	Ignores quadratic *synergy* costs; only linear assignment.
Combinatorial Auction	Very High	Approximate	Dynamic	Decentralized	IA-AFL frameworks [22]	Winner determination is NP-hard, but good heuristics exist.

Section 2: Workflow Routing and DAG Scheduling

This problem moves from *who* does *what* (assignment) to *when* they do it (sequencing). The core challenge in the ATLAS system is that its workflows are not static. The "dialectical chain" (Designer$\rightarrow$Critic$\rightarrow$Refactorer$\rightarrow$Validator) is a dynamic, quality-dependent, and potentially *cyclical* graph, not a simple Directed Acyclic Graph (DAG).

2.1 DAG Scheduling Fundamentals

Workflows are conventionally modeled as DAGs, where nodes are tasks (agents) and edges represent precedence constraints.[30]

- **Classical Methods:** These serve as essential baselines for any static sub-problems.
  - **Critical Path Method (CPM)/PERT:** Identifies the longest path (the "critical path") through the DAG, which defines the minimum possible project completion time (makespan).
  - **List Scheduling:** A family of effective heuristics. A "ready list" of tasks whose dependencies are met is maintained. An algorithm then repeatedly selects a task from this list based on a priority rule (e.g., shortest processing time, critical path priority) and assigns it to an available resource.
- **Relevance:** These methods are necessary for any *known, static* portions of a research plan (e.g., "Run Literature Scout, *then* in parallel run 3 Hypothesis Generators, *then* aggregate"). They are, however, insufficient for adaptive logic.

2.2 Adaptive and Confidence-Aware Routing

The "dialectical chain" presents a far more complex problem. The decision to route an artifact *back* to the Refactorer (creating a cycle) or *forward* to the Validator is not pre-defined; it is a *dynamic, probabilistic* decision based on an intermediate quality/confidence score.

This is a central topic in 2024/2025 agentic AI research, which is moving beyond static chains to dynamic routing.

- **STRMAC (State-Aware Routing Framework) (Nov 2025):** This SOTA framework proposes a lightweight router that adaptively selects the *single most suitable agent* at each step.[33] It explicitly encodes the "interaction history" (the current state) and "agent knowledge" (expertise) as separate embeddings.[33] The router, trained via contrastive learning, then matches the current state embedding to the best agent embedding (e.g., via cosine similarity) to make the routing decision.[36] This provides a formal mechanism for dynamic agent selection.
- **SelfOrg (Oct 2025):** This framework takes an even more dynamic approach, *constructing the DAG on the fly*.[32] Agents independently generate responses to a query. A graph is then dynamically constructed based on each agent's "Shapley value" (a game-theoretic measure of contribution), ensuring that information flows from high-contributing agents to others.[32] This is ideal for "swarming" on a complex problem.
- **GATES (May 2025):** This method [31] uses **Graph Attention Networks (GATs)** combined with an **Evolution Strategy** for *cost-aware dynamic workflow scheduling*. This is a full-blown ML-based scheduler that *learns* the optimal scheduling policy for dynamic,

DAG-based tasks.[31]

- **RL for Routing (Mar 2025):** This work [38] proposes a more direct RL approach. It uses an RL agent to *dynamically tune the weighting factors* of an adaptive routing algorithm (based on Dijkstra's). The weights are a function of context like task complexity, priority, and agent load.[38]

Stochastic Scheduling: The underlying theoretical field is stochastic scheduling, which deals with uncertainty in task durations or outcomes. For example, Stochastic Job Shop Scheduling (SJSSP) models processing times as random variables.[40] The ATLAS system's "confidence-aware" routing is a form of stochastic scheduling where the *graph structure itself* is stochastic.

2.3 AI Pipeline Orchestration Tools

A critical implementation decision is selecting the right tool to execute these workflows. The main contenders are traditional task orchestrators and newer agent-specific frameworks.

- **Traditional Orchestrators:**
  - **Airflow:** The enterprise incumbent. It is fundamentally *static*.[41] While DAGs are defined in Python, they are not dynamically modified at runtime. It is operationally complex, requiring multiple services.[42] It is *not* suitable for implementing dynamic, confidence-based loops.[41]
  - **Prefect / Dagster:** Modern, Python-native alternatives.[43] Their key advantage is support for *dynamic dataflows*.[41] Prefect, for example, allows simple Python functions to be "flows" and excels at dynamic event management and reacting to changing data states.[41]
- Architectural Synthesis (Orchestrator vs. Agent Framework):
  A common architectural mistake is to misuse these tools. An orchestrator like Airflow or Prefect is designed to schedule tasks, which are typically long-running, reliable, and have static dependencies. An agent framework like LangGraph is designed to schedule LLM-based actors, whose interactions are fast, cyclical, stateful, and dynamic.45
  The optimal architecture for ATLAS/TURING is therefore *hierarchical*:
  1. **Level 1 (Task Orchestrator):** Use **Prefect** (or Dagster) [43] to manage the high-level, mostly linear research plan (e.g., Task A: Run Project_Alpha, Task B: Run Project_Beta).
  2. **Level 2 (Agent Framework):** Task A internally invokes a workflow built with **LangGraph**.[45] This LangGraph workflow manages the complex, cyclical, confidence-based logic of the Designer$\rightarrow$Critic$\rightarrow$Refactorer loop.

This design uses the right tool for the right job, leveraging Prefect's dynamic task management [43] and LangGraph's explicit support for stateful, cyclical graphs.[45]

2.4 Research Gap: Optimizing the Dialectical Chain

While SOTA (2025) frameworks like STRMAC [33] and SelfOrg [32] provide powerful mechanisms for *dynamic agent routing* (i.e., "who goes next?"), a critical gap remains. There is no formal work on optimizing the *convergence* of a "dialectical chain" (Designer$\rightarrow$Critic$\rightarrow$Refactorer).

The problem is not just "who is next?" but "when do we stop?". The rise of "multi-agent validation" [46], "critic" agents [48], and "dialectical reasoning" frameworks [49] confirms this is a nascent, ad-hoc field. Agentic coding systems like ChatDev [52] implement this loop, but the decision to stop or continue is based on simple heuristics (e.g., "run for k cycles" or "stop when Critic score > 0.9"). This can lead to infinite refinement loops [46] or, conversely, premature stopping that yields a suboptimal result. This is a prime research opportunity, which will be detailed in Section 7.2.

Table 2.1: Orchestration Tools vs. Adaptive DAG Scheduling Requirements

Tool	Primary Use Case	Graph Type	State Management	Confidence-Based Routing	Recommended ATLAS Use
Airflow	Batch Task Orchestration	Static DAG [41]	Via Task Instances	No (Must be custom-built)	**Not Recommended** (Too rigid).
Prefect	Dynamic Task Orchestration	Dynamic Python Flows [41]	First-class object [41]	Yes (via Python logic)	**Level 1 (Task Orchestrator):** Triggering

					high-level research plans.
LangGraph	Agent Orchestration	Stateful, Cyclical Graph [45]	Explicit State Object [45]	**Native Support** (via conditional edges)	**Level 2 (Agent Framework):** Implementing the dialectical D-C-R loops.
AutoGen	Agent Orchestration	Conversational (Emergent) [54]	Message History [54]	Yes (via agent logic)	Alternative to LangGraph, but less structured and harder to optimize.
STRMAC	Agent Routing	Dynamic (Single-step) [33]	Encoded State Vector [36]	Yes (This is its purpose)	**Research Baseline:** A SOTA method to compare against for routing.

Section 3: Resource Allocation and Meta-Learning

This section addresses the problem of distributing finite computational resources (e.g., CPU, GPU, API credits) among the 40+ competing agents. This is an exploration-exploitation problem: should resources go to a "proven" high-performing Designer agent (exploitation), or to a new, experimental Refactorer agent to see if it's better (exploration)?

3.1 Multi-Armed Bandit (MAB) Algorithms

The canonical model for the exploration-exploitation dilemma is the Multi-Armed Bandit (MAB).[56] Each of the K agents is an "arm." Pulling an arm (assigning a task) yields a stochastic "reward" (e.g., a quality score from the Validator). The goal is to maximize the cumulative reward over a time horizon T.

- **Algorithms:**
  - **Upper Confidence Bound (UCB):** An "optimism in the face of uncertainty" algorithm. It computes a high-confidence upper bound for each arm's true mean reward and pulls the arm with the highest UCB.[29]
  - **Thompson Sampling (TS):** A Bayesian approach. It maintains a *probability distribution* (a "belief") over each arm's reward (e.g., a Beta distribution for binary rewards, Gaussian for continuous). At each step, it samples one value from each arm's distribution and pulls the arm with the highest sample.[56] Empirically, TS often demonstrates lower regret (better performance) than UCB.[56]
- **Contextual and Non-Stationary Bandits:** The simple MAB model is insufficient for the ATLAS system.
  1. **Non-Stationary MABs (NSMAB):** Agent performance is not static; it will change (e.g., after a prompt update). This is a non-stationary environment.[57] NSMAB algorithms (like Sliding-Window UCB or Discounted Thompson Sampling) are required, as they give more weight to recent data.
  2. **Contextual Bandits:** The reward of an agent (arm) depends on the *context* (e.g., task type, difficulty, associated artifacts). At each step t, a context vector x_t is observed.[29] Algorithms like LinUCB and Contextual Thompson Sampling [29] learn a function that maps contexts to rewards, allowing for generalized, intelligent decisions.

Recent SOTA (2023-2025) applications of MABs align perfectly with this, including dynamic RAG (where each retrieval method is an arm) [58], LLM-informed MABs for non-stationary environments [59], and secure VM allocation.[60]

3.2 Constrained Resource Allocation: Bandits with Knapsacks (BwK)

This is the precise formulation required for the ATLAS system. In the **Bandits with Knapsacks (BwK)** problem, pulling an arm (running an agent) yields not only a reward (quality) but also incurs a *cost* (e.g., API credits, GPU-seconds). The agent must maximize cumulative reward *without exceeding a total budget* (the knapsack).[61]

This is a highly active research area in 2024-2025:

- **SOTA Algorithm:** A key 2025 paper [61] proposes **LinCBwK9TS**, a new **Thompson Sampling** algorithm for **Linear Contextual Bandits with Knapsacks (CBwK)**. This work is motivated by the superior empirical performance of TS over UCB-based methods and provides a strong theoretical regret bound.[61] This algorithm is a perfect SOTA baseline for the ATLAS resource allocator.
- **Delayed Feedback:** The DORAL algorithm (2024) [63] specifically addresses the BwK problem where the reward feedback is *delayed*—which is highly relevant for research tasks that may take hours or days to validate.
- **Dynamic Constraints:** A 2025 paper [64] introduces "Budgeted UCB," an algorithm for environments where the *constraints themselves are dynamic* (e.g., a "decaying violation budget").

The current research frontier (2025) in this area is moving toward handling complex real-world scenarios like delayed feedback [63] and *bi-level* allocation (e.g., allocating budgets to *groups* of agents, then individuals) [66], which leads directly to the novel contribution in Section 7.3.

3.3 Meta-Learning for Algorithm Selection

The ATLAS system will not just *use* optimization; it *is* an optimizer. At different times, it may need to solve a QAP, a scheduling problem, or a resource allocation problem. **Meta-Learning**, or "learning to learn" [67], is a framework for selecting the *best* optimization algorithm for the *specific problem instance* at hand.

A 2024 survey [68] formally defines this field as **Meta-Black-Box-Optimization (MetaBBO)**. MetaBBO uses a meta-level learner (which can be an RL agent, a supervised model, or even an LLM) to automate algorithm design tasks, including:

1. **Algorithm Selection:** Choosing from a portfolio of solvers (e.g., "This QAP is small, use Gurobi; this one is large, use Tabu Search").[68]
2. **Algorithm Configuration:** Setting the hyperparameters of a chosen solver (e.g., "For this problem, set the TS tabu list size to 20").[68]
3. **Algorithm Generation:** The most advanced form, where the meta-learner *generates* a new, bespoke heuristic algorithm.[68]

SOTA (2024) work uses RL agents to *dynamically* control the parameters of evolutionary algorithms during their run.[72] The introduction of the MetaBox benchmark [73] in 2024 for MetaBBO-RL indicates this field is rapidly maturing. For ATLAS, a MetaBBO layer could learn a high-level policy for selecting its own coordination strategies.

3.4 Agent Tournaments and Performance Tracking

To optimize the 40+ agents over time, the system needs a mechanism to (1) evolve their configurations (prompts, models) and (2) measure their performance.

- **Population-Based Training (PBT):** This is a SOTA hyperparameter optimization (HPO) technique that is far more efficient than grid search.[74] PBT trains an entire *population* of agents (e.g., 20 versions of Designer) in parallel.[75] Periodically, it "exploits" by copying the weights and hyperparameters from the top-performing agents to the bottom-performing ones, and then "explores" by perturbing (mutating) those new hyperparameters.[76] The Ray Tune library has a mature, scalable PBT implementation.[76] Generalized PBT (GPBT) (2024) [78] is a recent extension.
- **Elo Rating System:** PBT requires a *fitness function* to identify the "top performers." For complex research outputs, a simple accuracy metric is insufficient. The SOTA solution for ranking models on open-ended tasks is the **Elo rating system**.[79] This system, famously used by Chatbot Arena [81], turns pairwise comparisons ("Is output A better than output B?") into a cardinal numeric score.[83]

Architectural Synthesis (PBT + Elo = Continual Improvement):
These two concepts can be combined to create a fully autonomous, continual improvement engine for the ATLAS agent population.

1. **Population:** The 40+ agents are managed as a population within Ray Tune's PBT scheduler.[76]
2. **"Battles":** As agents produce artifacts, the Validator and Critic agents are tasked with performing pairwise comparisons (e.g., Designer_v1.2's output vs. Designer_v1.3's output for the same task).
3. **Leaderboard:** These pairwise "votes" are continuously fed into an internal **Elo rating system** [80], maintaining a real-time leaderboard of all agent versions.
4. **Evolution:** At each PBT "exploit" step, the scheduler queries this Elo leaderboard to identify the "top performers," thus closing the loop.

This creates a robust, asynchronous, and continuous process for evolving and optimizing all agents in the system based on their demonstrated performance.

Section 4: Evolutionary Architecture Search and Auto-Discovery

This section addresses the most advanced coordination problem: moving from

hand-designing the 40-agent workflow to enabling the system to *automatically discover* superior coordination patterns. This treats the workflow architecture itself as a "genome" that can be optimized.

4.1 Evolutionary Workflow Optimization

This is the bleeding-edge SOTA (2025) in agent coordination. Instead of optimizing a *fixed* graph, these frameworks evolve the *graph structure itself*.

- **EvoFlow (Feb 2025):**
  - **Concept:** This framework [84] uses a niching-based evolutionary algorithm to optimize a *population* of diverse, heterogeneous agentic workflows *on the fly*.[84]
  - **Multi-Objective:** It frames the search as a multi-objective problem: *cost vs. performance*.[84] The output is not one "best" workflow, but a *Pareto-optimal set* (e.g., a cheap, fast workflow for simple tasks and an expensive, complex workflow for hard tasks).[85]
  - **Genome and Operators:** The workflow is the genome. Evolutionary operators are applied, including **LLM Mutation** (swap the model), **Prompt Mutation**, and **Operator Mutation** (change the agent/tool).[84] Selection is "tag-based" (finds relevant parents for crossover) and "niching-based" (maintains population diversity).[84]
  - **Implementation:** github.com/bingreeky/EvoFlow.[84]
- **EvoAgentX (July-Nov 2025):**
  - **Concept:** This is an open-source *platform* and *integrator* that automates multi-agent workflow generation from high-level goals.[86]
  - **Integrated Optimizers:** Its primary contribution is its "evolving layer," which integrates several SOTA optimization algorithms [86] to iteratively refine prompts, tools, and topologies:
    1. **TextGrad** (2024) [86]: For prompt optimization.
    2. **AFlow** (2024) [86]: For workflow graph optimization.
    3. **MIPRO** (2024) [86]: For prompt optimization.
  - **Performance:** The framework demonstrates significant performance gains on SOTA benchmarks like GAIA, MATH, and HotPotQA.[88]
  - **Implementation:** github.com/EvoAgentX/EvoAgentX.[87]
- **EvoAgent (NAACL 2025):**
  - **Concept:** A related framework [94] that proposes a generic method using evolutionary algorithms (mutation, crossover, selection) to *automatically extend* a specialized single-agent system into a more capable *multi-agent system*.[94] This is a powerful

"bootstrapping" mechanism.

4.2 Neural Architecture Search (NAS) for Agent Topologies

This approach adapts principles from Neural Architecture Search (NAS)—the field of auto-discovering optimal neural network architectures—to the problem of discovering optimal *agent network topologies*.

- **MASS (Multi-Agent System Search) (Feb 2025):**
  - **Concept:** A NAS-inspired optimization framework designed to automate MAS design by optimizing both *prompts and topologies* simultaneously.[95]
  - **Method:** It employs a 3-stage interleaved optimization process [96]:
    1. **Stage 1:** Block-level (local) prompt optimization.
    2. **Stage 2:** Workflow topology optimization (searching over the graph structure).
    3. **Stage 3:** Workflow-level (global) prompt optimization (refining prompts for the chosen topology).
  - **Key Finding:** The MASS paper [99] reveals that optimizing individual agent prompts *first* (Stage 1) is critical for success before attempting to optimize the large-scale topology (Stage 2).
- **NADER (CVPR 2025):**
  - **Concept:** This framework [100] presents a fascinating reversal: it uses a *multi-agent collaboration* of LLM agents (e.g., Proposer, Reflector) to *perform Neural Architecture Design*.[100]
  - **Representation:** Both MASS [99] and NADER [100] emphasize the critical importance of the *search space representation*. NADER advocates for a *graph-based representation* of the architecture, rather than raw code. This allows the optimizing agents to focus on high-level structural decisions without being "distracted by coding".[100]
- Differentiable Workflow Evolution:
  Making the (discrete) workflow graph evolution differentiable, to enable gradient-based optimization (a la DARTS 100), is a major challenge. The TextGrad framework 86, integrated into EvoAgentX, is a key step in this direction, as it proposes a method to "backpropagate" text-based feedback to optimize prompts. A fully gradient-based search over discrete agent topologies remains a significant open research problem.

4.3 Workflow Genome Representation and Constrained Evolution

- **Representation:** To evolve a workflow, it must be encoded as a "genome." Common representations include:
  - **Graph:** A formal graph with nodes (agents) and edges (connections).[100]
  - **Tree:** A prefix tree, as used in classical Genetic Programming (GP).[105]
  - **Script:** A coordination script (e.g., Python code) that is itself evolved.[106]
- **Operators:** Standard evolutionary operators are adapted: **Mutation** (add/remove agent, change connection, change agent's prompt [84]) and **Crossover** (combine sub-graphs from two parent workflows).[94]
- **Constrained Evolution (Critical Gap):** This is a critical limitation for the ATLAS system's safety. An unconstrained evolutionary algorithm will generate thousands of *invalid* or *unsafe* workflows (e.g., a Refactorer that writes directly to the final Archive without a Validator step).
  - **The Gap:** SOTA frameworks like EvoFlow [84] and MASS [95] optimize for performance and cost, but not for *hard safety constraints*.
  - **SOTA:** EvoAgent (NAACL 2025) [94] *mentions* assessing an agent's ability to "satisfy individual constraints," but the mechanism is not detailed. This field, which combines agent evolution with "Safe MARL" [108] and "Constrained Policy Optimization" [109], is wide open for novel contributions (see Section 7.4).

Table 4.1: Frameworks for Evolutionary Workflow Auto-Discovery (2025)

Framework	Optimization Goal	Method	Optimized Artifacts	Constraint Support	Key Contribution
EvoFlow [84]	Pareto Population (Cost vs. Perf.)	Multi-Objective Evolutionary [84]	Prompts, LLM Models, Operators	None (Penalty-based)	Evolves a *diverse population* of workflows, not just one.
EvoAgentX [86]	Single High-Performance	Integrator (Evolutionary) [86]	Prompts, Tools, Topologies [88]	Unclear (via optimizers)	An *integrator platform* that

	Workflow				bundles SOTA optimizers (TextGrad, AFlow, MIPRO).[86]
MASS [95]	Single High-Performance Workflow	NAS-Inspired Search [98]	Prompts and Topologies [98]	None (Unconstrained search)	A 3-stage search that interleaves prompt- and topology-optimization.[96]
EvoAgent [94]	Single High-Performance Workflow	Evolutionary [94]	Agent Settings / Team Structure	Mentioned [94]	Evolves a *single agent* into a *multi-agent system*.[94]

Section 5: Game Theory, Adversarial Optimization, and Robustness

This section addresses the crucial requirement of ensuring the 40-agent system is robust. This includes robustness to adversarial inputs, unexpected failures, and strategic agent behavior.

5.1 Min-Max and Bi-Level Optimization for Robustness

A robust system is one that performs well even in the *worst-case* scenario. This can be formally modeled as a **min-max optimization** problem: we want to find the workflow W that *minimizes* the *maximum* loss (or "regret") that an *adversary* A can inflict.

This is a formal **bi-level optimization** problem [110]:

- **Outer Loop:** $\min_{W \in \text{Workflows}}$ (Loss(W, Input^*))

- **Inner Loop:** $\max_{\text{Input}^* \in \text{AdversarialSet}}$ (Loss(W, Input^*))

The inner loop finds the worst-case input Input^* for a *given* workflow W, and the outer loop finds the workflow W that is the most robust to its *own* worst-case input.

A SOTA (Sep 2025) framework, **RobustFlow** [110], directly addresses this. It is a training framework that *leverages preference optimization (DPO)* to teach the workflow *generator* model to be *invariant to instruction variations* (e.g., paraphrasing).[110] It trains the model to learn a single, canonical workflow for a cluster of synonymous queries.[110]

However, RobustFlow makes the workflow *generation* robust to *input text variations*. It does not make the workflow *execution* robust to *agent failure* or *environmental shifts*. This remains a critical gap (see Section 7.5).

5.2 Adversarial Testing and Red-Teaming Frameworks

This is the practical implementation of the "inner loop" $\max(\text{Loss})$: how to find the "AdversarialSet." A "Red Team" subsystem must continuously attack the ATLAS system to find vulnerabilities.

Several SOTA (2024) open-source frameworks exist for this purpose:

- AgentSafety (Repo) [112]: This is a comprehensive GitHub repository from OSU-NLP-Group that curates research and tools for agent safety. It lists numerous attack frameworks, including:
  - AdvWeb: A black-box attack on VLM-based web agents.
  - RedAgent: A multi-agent system designed to *automatically find* jailbreak prompts.[112]
  - Environmental Injection Attacks (EIA): Attacks that inject malicious content into the agent's environment.[112]
- BrowserART (Oct 2024) [112]: The "Browser Agent Red teaming Toolkit".[112] This is a concrete *test suite* with 100 diverse harmful behaviors (sourced from HarmBench and AirBench) to test browser-based agents.[112]
- DeepTeam (2024) [113]: An open-source *LLM red teaming framework* for penetration testing. It is a practical toolkit that simulates 10+ attack methods (e.g., prompt injection, leetspeak, crescendo jailbreaking) and checks for over 40 vulnerabilities (e.g., bias, PII leakage, robustness).[113]
- AATMF (2024) [114]: The "Adversarial AI Threat Modeling Framework." This provides an OWASP-style *methodology* and taxonomy for testing AI systems, with 14 tactics like "Persona Override" (AT-001) and "Reasoning Exploitation" (T3).[114]

A powerful, novel architecture would unite these components: the *testing* frameworks (DeepTeam, RedAgent) [112] are used to *generate* the training data for the *adversary* in the *inner loop* of the *bi-level optimization* [111] that *designs* the robust workflow. This creates a self-improving, adversarially-trained system.

5.3 Game-Theoretic Coordination

When agents have different (potentially conflicting) objectives, their coordination can be modeled as a "game."

- **Robust MARL (2023-2024):** A significant body of research [115] focuses on "Robust MARL." A 2024 SOTA paper [118] explores **Distributionally Robust Markov Games (RMGs)**. In an RMG, each agent learns a policy that maximizes its *worst-case* performance when the environment deviates from the training distribution.[118] This is critical for ATLAS, ensuring its agents perform reliably when the "distribution" of research problems changes.
- **Stackelberg Games (2024):** This model is ideal for leader-follower dynamics.[119] A 2024 paper [120] applies "Stackelberg game-theoretic learning" to multi-robot collaborative assembly. This is a very strong analogue for the ATLAS Designer$\rightarrow$Critic relationship. The Designer (leader) makes a move (generates an artifact), and the Critic (follower) plays its best response (provides a critique). This can be used to find a "Stackelberg equilibrium," where the Designer anticipates the Critic's critique and produces a better initial artifact.
- **Mechanism Design:** A branch of game theory that *designs the rules of the game*. For ATLAS, this applies directly to the design of the **Auction-Based Task Allocation** (Section 1.3). The auction mechanism [22] is designed so that the self-interested, "bidding" behavior of the 40+ agents results in a globally efficient and truthful assignment.

Table 5.1: Adversarial Testing Frameworks for LLM-Agent Systems (2024)

Framework	Type	Attack Focus	Target Agent	Open Source
DeepTeam [113]	Toolkit	Jailbreaking, PII Leakage,	General LLM, Chatbot, RAG	Yes [113]

		Bias, Misinformation (40+ vulnerabilities)		
BrowserART [112]	Test Suite	Harmful Behaviors (100 tasks, e.g., web-based attacks)	Browser Agents	Yes (via AgentSafety) [112]
AATMF [114]	Methodology	Prompt Subversion, Linguistic Evasion, Memory Manipulation (14 tactics)	General AI Systems	Yes [114]
RedAgent [122]	Attack Generator	Context-aware Jailbreak Prompt Generation	Black-box LLMs	Yes (via AgentSafety) [122]

Part 2: Comparative Analysis of Frameworks, Tools, and Benchmarks

Section 6: Platforms, Libraries, and Testbeds

This section provides a comparative analysis of production-ready tools, research-prototype libraries, and benchmark datasets, offering recommendations for the "buy vs. build" decisions in the ATLAS/TURING architecture.

6.1 Multi-Agent Orchestration Platforms

The choice of orchestration framework is a fundamental architectural decision. The 2024-2025 landscape is dominated by a few key players.[54]

- **LangGraph (2024):**
  - **Architecture:** A library from LangChain for building *stateful, multi-actor applications*.[45] Its key feature is treating workflows as *cyclical graphs* rather than DAGs.[54]
  - **Pros:** Explicit, persistent state management (a central state object is passed and modified by each node). Native support for *cycles, branching, and human-in-the-loop*.[45] This is, by far, the best-suited framework for implementing the Designer$\rightarrow$Critic$\rightarrow$Refactorer loop.
  - **Cons:** Can be complex, requiring explicit definition of the graph, state schema, and conditional edges.[54]
- **Microsoft AutoGen (2023/2024):**
  - **Architecture:** A generic *multi-agent conversation* framework.[54] Agents (e.g., UserProxyAgent, AssistantAgent) coordinate by sending and receiving *asynchronous messages*.[54]
  - **Pros:** Highly flexible, good for "emergent" group chat behaviors, supports both autonomous and human-in-the-loop workflows.[54]
  - **Cons:** The "conversation-driven" flow can be chaotic, difficult to debug, and get stuck in loops.[54] State management is less explicit than in LangGraph, as it is embedded in the message history.
- **CrewAI (2024):**
  - **Architecture:** A *role-based* framework.[54] The developer defines Agents (with roles, backstories, and goals), Tasks (with descriptions), and a Crew that executes them using a Process.[54]
  - **Pros:** Extremely simple to get started, excellent for rapid prototyping, and enforces a clear, hierarchical structure.[54]
  - **Cons:** The default "sequential" process is far too rigid for the dynamic needs of ATLAS.[54] It is not designed for complex, graph-based, or cyclical logic.
- **Swarm (OpenAI, 2024/2025):**
  - **Architecture:** This is currently experimental and not a production-ready framework.[123] The name, along with related research into swarm intelligence [125] and decentralized systems (see Section 7.7), suggests a future direction, but it is not a tool that can be adopted today.

Recommendation: A hybrid architecture is required.

1. **Level 1 (Task Scheduling):** Use **Prefect** [43] for high-level, dynamic task orchestration (as discussed in Section 2.3).
2. **Level 2 (Agent Coordination):** Use **LangGraph** [45] to build the internal, stateful, cyclical workflows (e.g., the dialectical chains) that are *triggered* by Prefect.

6.2 Blackboard and Shared Memory Architectures

The 40+ agent system will be too large and complex to be managed by a single, monolithic LangGraph. A more scalable, flexible paradigm is the **Blackboard Architecture**, a classic AI pattern [126] being rediscovered for LLM agents.[127]

- **Concept:** Agents do not communicate directly (peer-to-peer) or via a rigid graph (orchestrator). Instead, they communicate *indirectly* by reading and writing to a central, structured data store (the "blackboard").[126]
- **Modern Relevance (2024/2025):**
  - MetaGPT (2024) [128] implements a "shared memory pool".[129]
  - A 2025 paper [126] explicitly revives the blackboard architecture, distinguishing it from simple shared memory: in a true blackboard system, agents *autonomously* monitor the board and *decide* to contribute when a relevant request is posted. This contrasts with a shared-memory system where a central coordinator still *assigns* tasks.[126]

This leads to a more sophisticated *hybrid architecture* recommendation:

- Use **LangGraph** for *small, well-defined, tactical* workflows (like the D-C-R loop) where the sequence is known but cyclical.
- Use a **Blackboard System** (which would need to be custom-built) for *high-level, strategic, emergent* coordination. For example:
  1. Literature Scout finishes its run and posts an "Artifact: New_Paper_Summary" to the blackboard.
  2. This post *triggers* all agents subscribed to that topic.
  3. A Hypothesis Generator autonomously pulls the summary, processes it, and posts a "Artifact: New_Hypothesis" to the board.
  4. This new post, in turn, triggers a Designer LangGraph team to spin up and begin working on the hypothesis.
     This architecture is decentralized, flexible (new agents can be added at runtime by just subscribing to topics), and robust to agent failure. This connects to the SwarmSys decentralized model 130 and is a key topic for a novel contribution (Section 7.7).

6.3 MARL Platforms

For any research involving learning-based coordination (e.g., Sections 7.2, 7.5, 7.6), a dedicated MARL stack is necessary.

- **RLlib (from Ray):**
  - **Features:** A highly scalable, framework-agnostic library for RL.[131] Its most critical feature for ATLAS is its built-in support for **hierarchical training**, with examples of a "manager" agent calling "worker" agents.[132] It integrates directly with PettingZoo and OpenSpiel.[132]
  - **Cons:** Notoriously complex and difficult to debug.[134]
- **PettingZoo:**
  - **Features:** This is *not* a training library, but the *de facto standard API* for multi-agent environments (it is the multi-agent version of Gymnasium).[134]
- **OpenSpiel (from DeepMind):**
  - **Features:** A framework for RL in *games*.[135] Its primary strength is its implementation of game-theoretic concepts, which is ideal for the research in Section 5.3 (e.g., Stackelberg games).[133]

Recommendation: The MARL stack is clear: (1) Define the 40-agent ATLAS environment using the **PettingZoo** API.[134] (2) Use **RLlib** as the training backend, leveraging its hierarchical agent support.[132]

6.4 Optimization Libraries (OR vs. ML)

- **OR Solvers (for QAP, Scheduling):**
  - Google OR-Tools: **Best open-source choice.** It is fast, Python-native, and has excellent, mature solvers for Constraint Programming (CP-SAT), scheduling (JSSP) [28], and routing (VRP), which are close cousins to QAP.[10]
  - Gurobi / CPLEX: **Best commercial choice.** These are the market leaders, known for "blazing speed" in solving large, complex MIPs (like QAP).[10] They are available for free via academic licenses.
- **ML Solvers (for HPO, Evolution):**
  - Optuna: **Best for HPO.** A SOTA hyperparameter optimization framework that uses Bayesian methods (TPE).[137] It also includes samplers for evolutionary algorithms like CMA-ES.[139]
  - Ray Tune: **Best for PBT.** The best-in-class, distributed implementation of Population-Based Training [76], which is essential for the PBT+Elo agent evolution

concept (Section 3.4).
    - DEAP: **Best for pure Genetic Programming.** A classic, flexible framework for evolving *code* or *trees*.[105] This would be the library to use for the *Safe-EvoFlow* research gap (Section 7.4).

6.5 Benchmark Datasets

Validating novel research requires SOTA benchmarks.

- **Classical Benchmarks:** QAPLIB [2] (for assignment), TSPLIB (for routing), and JSSP Datasets [28] (for scheduling). These are necessary to validate any new *solvers* (e.g., the OC-QAP in Section 7.1).
- **SOTA Agent Workflow Benchmarks (2023-2025):** These are essential for validating the *agent architecture* itself.
    - AgentBench (2023) [141]: Evaluates reasoning and decision-making in 8 diverse environments (OS, DB, Knowledge Graph, Web).[141]
    - WebArena (2023) [141]: Focuses on realistic, multi-step web-based tasks (e-commerce, collaborative code development).[141]
    - GAIA (2023) [141]: The SOTA benchmark for "General AI Assistants." Requires long-horizon reasoning, multi-step tool use, and multimodality.[141] This was used to validate EvoAgentX.[92]
    - ToolBench / ToolLLM (2023) [141]: The largest benchmark for *tool use* and API calls (16,000+ APIs).[141]
    - ItineraryBench (2025) [16]: A new benchmark from Feb 2024 (v2 Jan 2025) for *dynamic, complex, multi-step* planning tasks.[16]

Key Gap ("ATLAS-Bench"):
There is no public benchmark for evaluating a multi-agent system on end-to-end autonomous scientific research. Anthropic has published a blog post on its internal multi-agent research system 143, but no public testbed exists.
The *first publication* from the ATLAS project could be the definition and open-sourcing of **"ATLAS-Bench: A Benchmark for Evaluating Autonomous Multi-Agent Scientific Research Systems."** This would be a high-impact contribution, defining a new testbed for the entire field.

Table 6.1: Multi-Agent Orchestration Framework Comparison (2024/2025)

Framework	Core Architecture	State Management	Key Pro	Key Con	Maintenance Status
LangGraph [45]	Stateful, Cyclical Graph	Explicit State Object	Best for loops, branching, state	More complex "boilerplate" [123]	Actively Maintained (2025) [54]
AutoGen [54]	Conversational (Async)	Message History	Flexible, "emergent" chat	Unstructured, can be chaotic [123]	Actively Maintained (2025) [54]
CrewAI [54]	Role-Based	Task-Based	Very simple, rapid prototyping [55]	Too rigid, default sequential [54]	Actively Maintained (2025) [54]
MetaGPT [128]	Blackboard-like	Shared Memory Pool [129]	Structured, role-based workflow	Less flexible than LangGraph	Actively Maintained (2024) [128]
OpenAI Swarm [124]	Decentralized (Swarm)	Unknown	(Theoretical) Scalability, Robustness	Experimental, not production-ready [123]	Experimental (2024) [123]

Table 6.2: Optimization and MARL Library Ecosystem for ATLAS

Library	Problem Type	License	Recommended Use Case in

			ATLAS
Gurobi [10]	MIP / QAP / Scheduling	Commercial (Free Academic)	**(If licensed):** SOTA solver for static QAP/Scheduling baselines.
Google OR-Tools [28]	CP / MIP / Scheduling / Routing	Apache 2.0	**(Default):** Open-source solver for assignment and scheduling heuristics.
Optuna [137]	Hyperparameter Optimization (HPO)	MIT	General HPO for all models and agents (e.g., prompts, temperatures).
Ray Tune [138]	Scalable HPO / PBT	Apache 2.0	**Core component:** Running the PBT+Elo system for agent evolution.[76]
DEAP [105]	Genetic Programming (GP)	LGPL-3.0	**Research component:** Backend for Safe-EvoFlow (Gap 7.4).
PettingZoo [135]	MARL Environment API	MIT	**Core MARL component:** Defining the 40-agent system as an RL environment.
RLlib (Ray) [132]	MARL Training	Apache 2.0	**Core MARL component:** Training algorithms for H-MARL (Gap

			7.6).

Part 3: Research Gaps and Novel Contribution Opportunities

This section synthesizes the preceding analysis to identify seven high-impact, publishable research contributions. These are designed to be novel in the 2024-2025 research landscape and directly address the unique challenges of the ATLAS/TURING system.

Table 7.1: Matrix of Novel Research Contributions

Gap ID	Research Gap	Problem Domain	Novel Contribution	Key Baselines	Target Venue	Feasibility
7.1	**Contextual QAP**	Agent-Task Assignment	**OC-QAP:** GNN-based heuristic solver	Static QAP [2], MAB [29], STRMAC [33]	AAAI, ICLR	Medium
7.2	**Dialectical Chain**	Workflow Scheduling	**SSP-QC:** Model loop as Stochastic Shortest Path	Fixed-loop, STRMAC [33]	AAMAS, NeurIPS	High
7.3	**Hierarchical BwK**	Resource Allocatio	**HBwK:** Two-level (Team/Ag	Flat BwK-TS	ICML, AISTATS	High

		n	ent) bandit	[61], DORAL [63]		
7.4	**Constrained Evolution**	Architecture Search	**Safe-EvoFlow:** EvoFlow + constrained operators	EvoFlow [84], MASS [95], Safe MARL [108]	GECCO, IEEE TEVC	Medium
7.5	**Adversarial Robustness**	Robust Workflow Design	**M2-WO:** Bi-level min-max (Evo vs. RedAgent)	RobustFlow [110], DeepTeam [113]	NeurIPS, ICLR	Low-Medium
7.6	**Meta-Control**	MARL Orchestration	**H-MARL:** Meta-controller sets sub-agent goals	Flat MARL (RLlib) [132], EvoAgentX [86]	AAMAS, CoRL	Medium
7.7	**Decentralized Swarm**	Orchestration Architecture	**Stigmergy-Blackboard:** Emergent coordination	MetaGPT [130], LangGraph [45], SwarmSys [130]	AAMAS, TAAS	Medium

---

7.1 Gap: Contextual QAP for Dynamic Agent Assignment

Existing Work: The Quadratic Assignment Problem (QAP) [1] is a powerful model for static assignment with synergies. However, its cost matrices are static. For dynamic assignment, the field has bifurcated: (1) simplified models like Contextual Multi-Armed Bandits (MABs) [29], which are dynamic but ignore quadratic synergies, or (2) dynamic routing frameworks like

TDAG [16] and STRMAC [33], which are ad-hoc and do not formally optimize global assignment cost.

Specific Limitation: There is no unified framework for **Online Contextual QAP (OC-QAP)**, where the full $N \times M$ cost matrix $C_{ij}(s_t)$ (cost of agent i on task j) and the $N \times N$ flow matrix $F_{ik}(s_t)$ (synergy of agent i and k) are both functions of the dynamic system state s_t (e.g., agent load, agent confidence, task features). Solving a full QAP at every time-step t is intractable.

Novel Contribution Proposal: "A Deep Learning Framework for Online Contextual Quadratic Assignment (OC-QAP) in Multi-Agent Systems."

Formulation: This contribution frames the OC-QAP as a *learned heuristic* problem, bridging ML4CO [11] and dynamic assignment.

1. **State Representation:** At time t, the system state s_t (agents, tasks, context) is represented as a heterogeneous graph.
2. **Cost Prediction:** A Graph Neural Network (GNN) model G_{θ} is trained. It takes the state graph s_t as input and *predicts* the full set of QAP cost matrices: $(C(s_t), F(s_t)) = G_{\theta}(s_t)$.
3. **Fast Assignment:** These predicted, context-aware cost matrices are fed into a *fast* (but sub-optimal) QAP heuristic solver (e.g., Tabu Search from QAPSolver [4] or a 2-Opt heuristic) to find the ϕ_t assignment for the current step.
4. **Training:** The GNN G_{θ} is trained via Reinforcement Learning. The "action" is the predicted cost matrix. The "reward" is the *downstream* task quality or system throughput achieved by the assignment ϕ_t. The GNN learns to *predict cost matrices that lead to good final solutions*.

Expected Novelty: This framework is the first to bridge the gap between static, high-synergy QAP and dynamic, low-synergy contextual MABs. It learns the *problem context* itself, rather than just learning a policy to *solve* a static problem.

Validation:

- **Baselines:** (1) Static QAP (solved once at $t=0$). (2) A greedy "best-agent" assignment (a contextual MAB [29]). (3) A full "flat" MARL policy.[132] (4) STRMAC [33] as a SOTA routing baseline.
- **Benchmark:** Run on a modified ItineraryBench [17] or a new, custom ATLAS-Bench.

Target Venue: AAAI (bridges AI and OR), ICLR (representation learning for optimization).

Feasibility: Medium. Requires deep expertise in both GNNs and combinatorial optimization.

7.2 Gap: Confidence-Aware DAG Scheduling (The Dialectical Chain)

Existing Work: SOTA routing frameworks like STRMAC [33] and SelfOrg [32] are "myopic"—they pick the best *next* agent. The "dialectical" pattern (Designer$\rightarrow$Critic$\rightarrow$Refactorer) is ubiquitous [46], and "Critic" [48] or "Dialectical Reasoning" [49] frameworks are emerging.

Specific Limitation: These frameworks do not formally optimize the *convergence* of the dialectical loop. The key decision ("Do we refactor again, or stop and validate?") is made by a simple heuristic (e.g., "stop at score > 0.9"). This can lead to infinite loops [46] or, more likely, premature stopping that results in a suboptimal artifact.

Novel Contribution Proposal: "Optimizing Dialectical Agent Workflows as a Stochastic Shortest Path (SSP) Problem with Quality Gates."

Formulation: Model the D-C-R loop as a Markov Decision Process (MDP), specifically a Stochastic Shortest Path (SSP) problem.

- **States (S):** A tuple representing the artifact's state: (v, q), where v is the artifact version (e.g., Code_v2) and $q \in $ is the Critic's last quality/confidence score.
- **Actions (A):** A set of actions available from any state: (Refactor, Validate, Stop_and_Archive).
- **Transitions (P):** A (learned) probabilistic transition function $P(s' \mid s, a)$. For example, $P((v+1, q') \mid (v, q), \text{Refactor})$ is the probability distribution over the *next* quality score q' given the current score q and the "Refactor" action. This model can be learned from execution history.
- **Costs (C):** A cost $C(s, a)$ associated with each action (e.g., Refactor cost = 10 API credits; Validate cost = 5).
- **Goal State:** Any state where $a = \text{Stop\_and\_Archive}$.
- **Objective:** Find an optimal policy $\pi^*: S \rightarrow A$ that minimizes the expected total cost to reach a goal state, while maximizing the final quality q at that state. This can be formulated as a standard SSP, solvable with Value Iteration or Q-Learning. The resulting policy π^* *is* the optimal confidence-aware scheduler.

Expected Novelty: This is the first formal, optimization-based model of the "dialectical agent loop," a ubiquitous but poorly understood pattern in agentic AI. It replaces a fragile heuristic with a provably optimal policy.

Validation:

- **Implementation:** Implement this policy as the conditional router in a LangGraph.[45]
- **Baselines:** (1) A fixed-loop policy (e.g., "always refactor $k=3$ times"). (2) A naive threshold policy ("stop when $q > 0.9$"). (3) An evolutionary baseline from EvoAgentX.[86]

- **Metrics:** Measure (Final Quality, Total Cost) pairs and show the SSP policy dominates the baselines on the Pareto front.

Target Venue: AAMAS (Autonomous Agents and Multiagent Systems), NeurIPS (RL Applications).

Feasibility: High. The formulation is clear, builds on standard MDP/RL theory, and is highly practical.

7.3 Gap: Constrained Hierarchical Bandits for Agent Resource Allocation

Existing Work: Contextual Bandits with Knapsacks (BwK) [61] are the correct *class* of models. The 2025 LinCBwK9TS algorithm [61] is a SOTA *implementation*. Nascent 2025 preprints are just beginning to explore *bi-level* contextual bandits [66] and delayed feedback.[63]

Specific Limitation: Standard BwK algorithms [61] treat all 40+ agents as a "flat" set of K arms. This is inefficient and ignores the known, exploitable *hierarchical structure* of the ATLAS agents (e.g., the "Coding" team, the "Validation" team, the "Literature" team). The bi-level work [66] is a starting point but is not fully developed for this team-based structure.

Novel Contribution Proposal: "Hierarchical Bandits with Knapsacks (HBwK): A Two-Level Approach for Team-Aware Multi-Agent Resource Allocation."

Formulation: Design a two-level MAB system based on the principles of hierarchical RL.

1. **Level 1 (Group-MAB):** This is a high-level **BwK** problem. The "arms" are the G *agent groups* (e.g., "Designers", "Critics", "Scouts"). This MAB learns to allocate the *total system budget* (e.g., 1000 GPU-hours) among these groups. Its reward is the *aggregate quality* produced by the group.
2. **Level 2 (Agent-MABs):** This is a set of G *independent* **Contextual BwK** problems (one for each group). Each group's MAB solves a *local* allocation problem (e.g., "How does the 'Designers' group spend its 200 GPU-hour budget (from Level 1) among its 3 individual agents, given the current task context x_t?").

Expected Novelty: This structure allows the system to learn *both* inter-group synergies (e.g., "we get more value by giving more budget to Critics right now") and intra-group performance (e.g., "Agent Critic-A is outperforming Critic-B on this task type"). It is a clear, formal extension of BwK [61] and bi-level bandits [66] that is highly practical for any large-scale, team-based agent system.

Validation:

- **Implementation:** Implement using Contextual Thompson Sampling at both levels, building on the SOTA LinCBwK9TS.[61]
- **Baselines:** (1) A "flat" CBwK (all 40 agents in one pool). (2) A static, non-bandit allocation (e.g., equal budget per group). (3) A non-hierarchical, "per-agent" CBwK.
- **Metrics:** Measure cumulative reward (quality) and regret vs. total budget. Hypothesize that HBwK learns faster (lower regret) due to its superior structural prior.

Target Venue: ICML (International Conference on Machine Learning), AISTATS (AI and Statistics).

Feasibility: High. This is a very strong, clear, and publishable extension of existing SOTA bandit theory.

7.4 Gap: Constrained Evolutionary NAS for Safe Agentic Workflows

Existing Work: SOTA (2025) frameworks like EvoFlow [84], EvoAgentX [86], and MASS [95] are powerful tools for *auto-discovering* high-performance agent workflows.

Specific Limitation: These frameworks optimize primarily for performance and cost.[84] They lack a formal mechanism for enforcing hard *safety or validity constraints*. For example, the ATLAS system may have non-negotiable rules: "A Refactorer output *must* be followed by a Validator before Code_Archive," or "A Hypothesis *must* be logged by *both* Literature_Scout *and* Validator." An unconstrained evolution will produce many high-performing but *unsafe* or *invalid* workflows. EvoAgent [94] mentions constraints, but the mechanism is unclear and not the focus.

Novel Contribution Proposal: "Safe-EvoFlow: Constrained Evolutionary Optimization for Verifiable Agentic Workflows."

Formulation: This contribution extends an evolutionary framework (like EvoFlow [84]) with formal principles from Constrained Optimization (CO) and Genetic Programming (GP).

1. **Representation:** Encode the workflow as a graph genome [100], as done in NADER.[100]
2. **Constraints:** Define a formal *constraint language* for agent workflows. This could use Linear Temporal Logic (LTL) or simple graph-path rules (e.g., ALWAYS (Refactor -> (NOT Archive UNTIL Validator))).
3. **Constrained Operators:** Design *constraint-preserving* or *constraint-aware* genetic operators (mutation/crossover).[84]
    - **Repair:** An invalid offspring (e.g., a mutation creates a Refactor -> Archive edge) is

"repaired" by an algorithm to make it valid (e.g., by inserting a Validator node).
    - **Safe Operators:** A "Safe Add Edge" operator *refuses* to create a connection that violates a constraint.
4. **Multi-Objective Goal:** Use a Multi-Objective Evolutionary Algorithm (MOEA) like NSGA-II to optimize a vector: (Objective 1: Performance, Objective 2: Cost, Objective 3: Constraint_Violation_Penalty). This allows the system to find a Pareto front of solutions that trade off performance with constraint satisfaction, or to enforce constraints strictly (infinite penalty).

Expected Novelty: This is the first application of formal "safe evolution" / "safe genetic programming" to the *architecture* of LLM agent workflows. It moves beyond "safe MARL" (which optimizes a policy for a *fixed* architecture) [108] to optimizing the *architecture itself* for safety.

Validation:

- **Benchmark:** Define a set of 10-15 hard safety constraints for the ATLAS-Bench (from Gap 6.5).
- **Baselines:** Run standard EvoFlow [84] and MASS.[95]
- **Metrics:** Show that Safe-EvoFlow finds a Pareto frontier where 100% of solutions are valid. Measure the performance/cost of the *best valid* solution found by the baselines vs. the Safe-EvoFlow frontier.

Target Venue: GECCO (Genetic and Evolutionary Computation Conference), IEEE Transactions on Evolutionary Computation (TEVC).

Feasibility: Medium. Requires deep expertise in genetic programming and constrained optimization, but the path is clear.

7.5 Gap: Adversarial Bi-Level Optimization for Agent Failure

Existing Work: Robustness is a major 2025 theme. RobustFlow (Sep 2025) [110] uses bi-level optimization to make workflow *generation* robust to *input instruction* variations. Adversarial *testing* frameworks (DeepTeam [113], BrowserART [112]) are for *post-design* validation.

Specific Limitation: No existing framework uses bi-level optimization [111] to design workflows that are robust to *agent failure* or *adversarial agent behavior* during *execution*. This includes a "lazy" Critic that passes all work, a "malicious" Refactorer that injects bugs, or a Literature Scout that simply dies (returns None).

Novel Contribution Proposal: "M2-WO: Min-Max Multi-Agent Workflow Optimization against

Adversarial Agent Failure."

Formulation: This is a true "adversarial training" for workflow architectures, formulated as a bi-level min-max game [111]:

- **Outer Loop (Proposer / Minimizer):** An optimizer (e.g., EvoFlow [84] or MASS [95]) proposes a workflow architecture W. Its goal is to $\min_{W}$ (Loss(W)).
- **Inner Loop (Adversary / Maximizer):** A "Red Team" policy π_{adv} tries to *break* W. This adversary can be a trained RL agent (like RedAgent [112]).
  - **Action Space:** $\text{Action} \in \{\text{Corrupt}(\text{Agent}_i, \text{mode}), \text{Delay}(\text{Agent}_j), \text{Kill}(\text{Agent}_k)\}$.
  - **Objective:** The adversary learns $\pi_{\text{adv}}^* = \arg\max_{\pi}$ (Final_Error(W, π)).
- **Fitness:** The Outer Loop's fitness for W is its *worst-case* performance: the minimized error from the inner loop's π_{adv}^*.

Expected Novelty: This system will learn to discover architectures that are inherently robust. For example, it might evolve *redundancy* (e.g., "always run two Critics in parallel and check for consensus"), *self-healing* (e.g., "evolve a 'meta-agent' that re-routes around a node that returns None"), or *stronger validation* (e.g., "evolve a workflow where the Validator is given more compute resources").

Validation:

- **Baselines:** (1) RobustFlow [110] (robust to input, not failure). (2) A non-robust EvoFlow baseline.[84]
- **Experiment:** After training, take the best workflows from all baselines. Perform "ablation attacks" (e.g., kill 1, 2, 3 agents at random).
- **Metrics:** Show that M2-WO produces workflows with significantly lower performance degradation when agents fail.

Target Venue: NeurIPS (Adversarial ML, Optimization), ICLR.

Feasibility: Low-to-Medium. This is a very ambitious, high-risk, high-reward, PhD-level project. The computational cost of the inner loop (training an adversary π_{adv}^* for *every* candidate workflow W) is immense.

7.6 Gap: Hierarchical MARL for Meta-Control

Existing Work: MARL frameworks (RLlib [132], PettingZoo [135]) provide the tools. RLlib has nominal support for "hierarchical training".[132] However, most H-MARL is applied to robotics

(e.g., "walk" sub-policy, "navigate" meta-policy). Separately, EvoAgentX [86] and MASS [95] use *external, offline* optimizers to *design* the workflow.

Specific Limitation: There is a lack of *online, two-level* RL systems where a *meta-controller* agent learns to *orchestrate* a team of *sub-agents* that are *also* learning. Applying H-MARL to *abstract agent coordination* is a novel frontier. 2025 preprints [145] explicitly identify hierarchical control and meta-learning as a key unexplored direction for multi-agent coordination.

Novel Contribution Proposal: "A Hierarchical MARL Framework for Meta-Control of Agentic Research Teams."

Formulation: A two-level H-MARL system, implemented using RLlib's hierarchical agent support [132]:

1. **Meta-Controller (Level 1):** A single RL agent that operates at a slow timescale (e.g., one action per research "stage").
    - **State:** The overall research progress (e.g., "Hypothesis A: validated; Hypothesis B: pending").
    - **Actions:** *Strategic* decisions (e.g., "Assign Team_Coding to Hypothesis_A," "Set objective for Team_Coding to 'Maximize_Readability'").
2. **Sub-Agents (Level 0):** Multiple MARL teams (e.g., Team_Coding = {Designer, Critic, Refactorer}). These agents operate at a fast timescale.
    - **State:** The local artifact (e.g., Code_v2).
    - **Action:** Execute the task (e.g., Critic runs).
    - **Reward:** Their reward is shaped by the *meta-controller's objective* (e.g., a "Readability_Score" from the Validator).

Expected Novelty: This directly implements a "manager" agent that learns how to *manage* other agents. It bridges the gap between EvoAgentX [86] (offline design) and RLlib [132] (flat MARL), creating an *online, adaptive* hierarchy that can respond to changing research priorities.

Validation:

- **Implementation:** Build in RLlib [132] with the environment defined in PettingZoo.[134]
- **Baselines:** (1) A "flat" 40-agent MARL system. (2) A hand-crafted, non-learning script (the STRMAC [33] or EvoAgentX [86] winner). (3) A non-learning meta-controller (e.g., round-robin).
- **Metrics:** Show that H-MARL achieves higher long-term reward (better research output) and adapts more quickly to *dynamic* changes in the meta-task (e.g., "a new high-priority paper was just published").

Target Venue: AAMAS, CoRL (Conference on Robot Learning - for the H-MARL theory).

Feasibility: Medium. H-MARL is notoriously difficult to get to converge, but the RLlib tooling

provides a strong foundation.

7.7 Gap: Decentralized Swarm Orchestration via Stigmergy

Existing Work: Most popular agent frameworks are *centralized* or *graph-based*. MetaGPT [128], LangGraph [45], and AutoGen [55] all rely on explicit message passing or a pre-defined graph structure.[130] Centralized orchestration is a bottleneck and a single point of failure.

Specific Limitation: The concept of **stigmergy**—indirect coordination via modifications to a shared environment [148]—is underdeveloped for LLM agents.

- **SwarmSys (Oct 2025):** This is the first SOTA paper [149] to propose a "decentralized swarm-inspired" framework.[130] However, it uses specific, pre-defined roles ("Explorer," "Worker," "Validator").[130]
- **Blackboard (2025):** The more *general* blackboard concept [126] (where *any* agent can subscribe to *any* task) has been proposed but not, to our knowledge, implemented and benchmarked against centralized orchestrators.

Novel Contribution Proposal: "Stigmergic Coordination of LLM Agents via a Unified Blackboard Architecture."

Formulation: Design and implement a pure *blackboard system* [126] as a general-purpose coordination mechanism.

1. **Blackboard:** A structured, shared database (e.g., Redis, MongoDB) where tasks and artifacts are posted with "Topics" or "Tags" (e.g., Topic:New_Hypothesis, Status:Pending_Review).
2. **Agents:** 40+ fully autonomous, "daemon-like" agents. Each agent's policy is defined by:
    - **Subscription:** The set of topics/tags it *listens* to (e.g., Critic subscribes to Status:Pending_Review).
    - **Action:** When triggered, it pulls the artifact, processes it, and *posts a new artifact* back to the board (e.g., Topic:New_Hypothesis, Status:Review_Complete, Score:0.8).
3. **Coordination:** Coordination is *emergent* and *indirect*.[148] There is no central orchestrator.
4. **Optimization:** The "optimization problem" is now to (a) learn the optimal *subscription* policy for each agent and (b) learn the optimal *prioritization* policy (if 10 tasks are on the board, which does an agent pick?).

Expected Novelty: This contrasts *directly* with the centralized, fixed-pipeline approach of MetaGPT [130] and the graph-based approach of LangGraph.[45] It is a completely different *paradigm* for multi-agent coordination, based on swarm intelligence.[152]

Validation:

- **Implementation:** Build the blackboard framework (this is a significant engineering task).
- **Baselines:** (1) A rigid, pre-defined LangGraph [45] workflow. (2) MetaGPT.[130] (3) The SwarmSys framework.[130]
- **Hypothesis:** The stigmergic system will be (a) slower for simple, linear tasks but (b) far more *robust* to agent failure (as any available Critic can pick up a Pending_Review task) and (c) more *flexible* (new agents can be added at runtime without *any* system redesign).

Target Venue: AAMAS, ACM Transactions on Autonomous and Adaptive Systems (TAAS).

Feasibility: Medium. Requires building a custom framework, but the concepts are well-established in classical AI.

Concluding Remarks and Architectural Synthesis

The analysis presented in this report provides a comprehensive roadmap for both the *implementation* and *research publication* phases of the ATLAS/TURING project. The 40+ agent system is not a single optimization problem but a *hierarchy* of them, requiring a hybrid, multi-layered architecture.

The recommended SOTA architecture synthesizes the findings as follows:

- **Task Orchestration: Prefect** [43] manages high-level, business-logic tasks (e.g., "start research project").
- **Strategic Coordination:** A **Blackboard System** [126] (as proposed in Gap 7.7) manages high-level, event-driven coordination between agent *teams* in a flexible, decentralized manner.
- **Tactical Coordination: LangGraph** [45] is used by individual teams to execute their internal, stateful, and *cyclical* logic (e.g., the Designer-Critic loop, formally optimized as an SSP per Gap 7.2).
- **Resource Allocation:** A **Hierarchical Bandits with Knapsacks (HBwK)** system (per Gap 7.3) manages the compute/API budget, allocating it first to *teams* and then to *individual agents*.[66]
- **Agent Performance Evolution:** A **Population-Based Training (PBT)** scheduler from Ray Tune [76] continuously evolves agent hyperparameters (prompts, models), using an internal **Elo Rating System** [81] (fed by Validator votes) as its fitness function.
- **Architecture Evolution:** A **Safe-EvoFlow** optimizer (per Gap 7.4) runs in the background, evolving the *topologies* of the LangGraph and Blackboard-subscription policies to discover better coordination patterns.[84]

- **System Robustness:** A **DeepTeam**-powered [113] "Red Team" subsystem continuously generates attacks, serving as the inner loop for an **M2-WO** bi-level optimizer (per Gap 7.5) to ensure the evolved architectures are robust to failure.

This architecture is modular, adaptive, and self-optimizing. Each component is grounded in SOTA 2024/2025 research and provides a direct path to a novel, publishable contribution.

Works cited

1. arXiv:2503.09676v1 [math.OC] 12 Mar 2025, accessed November 14, 2025, https://arxiv.org/pdf/2503.09676
2. New Benchmark Instances for the QAP and the Experimental Analysis of Algorithms - Metaheuristics Network, accessed November 14, 2025, https://www.metaheuristics.org/downloads/StuFer04.EvoCOP.pdf
3. [2007.14885] Performance Analysis of Meta-heuristic Algorithms for a Quadratic Assignment Problem - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2007.14885
4. TzeLun/QAPSolver: A simple Quadratic Assignment ... - GitHub, accessed November 14, 2025, https://github.com/TzeLun/QAPSolver
5. A comparison of different metaheuristics for the quadratic assignment problem in accelerated systems - ResearchGate, accessed November 14, 2025, https://www.researchgate.net/publication/347842280_A_comparison_of_different_metaheuristics_for_the_quadratic_assignment_problem_in_accelerated_systems
6. A Comparative Study of Meta-heuristic Algorithms for Solving Quadratic Assignment Problem - ResearchGate, accessed November 14, 2025, https://www.researchgate.net/publication/264084926_A_Comparative_Study_of_Meta-heuristic_Algorithms_for_Solving_Quadratic_Assignment_Problem
7. Quadratic programming solvers in Python with a unified API - GitHub, accessed November 14, 2025, https://github.com/qpsolvers/qpsolvers
8. qpsolvers/qpbenchmark: Benchmark for quadratic programming solvers available in Python, accessed November 14, 2025, https://github.com/qpsolvers/qpbenchmark
9. Supported solvers - qpsolvers 4.8.1 documentation, accessed November 14, 2025, https://qpsolvers.github.io/qpsolvers/supported-solvers.html
10. MIP Solvers Unleashed: A Beginner's Guide to PuLP, CPLEX ..., accessed November 14, 2025, https://medium.com/operations-research-bit/mip-solvers-unleashed-a-beginners-guide-to-pulp-cplex-gurobi-google-or-tools-and-pyomo-0150d4bd3999
11. Thinklab-SJTU/awesome-ml4co: Awesome machine learning for combinatorial optimization papers. - GitHub, accessed November 14, 2025, https://github.com/Thinklab-SJTU/awesome-ml4co
12. [2402.01968] A Survey on Context-Aware Multi-Agent Systems: Techniques, Challenges and Future Directions - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2402.01968

13. A Comprehensive Survey on Context-Aware Multi-Agent Systems: Techniques, Applications, Challenges and Future Directions - arXiv, accessed November 14, 2025, https://arxiv.org/html/2402.01968v2
14. [2312.15600] Context-aware Communication for Multi-agent Reinforcement Learning - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2312.15600
15. [2412.17964] Dynamic Multi-Agent Orchestration and Retrieval for Multi-Source Question-Answer Systems using Large Language Models - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2412.17964
16. [2402.10178] TDAG: A Multi-Agent Framework based on Dynamic Task Decomposition and Agent Generation - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2402.10178
17. TDAG: A Multi-Agent Framework based on Dynamic Task Decomposition and Agent Generation - arXiv, accessed November 14, 2025, https://arxiv.org/html/2402.10178v2
18. Advancing Agentic Systems: Dynamic Task Decomposition, Tool Integration and Evaluation using Novel Metrics and Dataset - arXiv, accessed November 14, 2025, https://arxiv.org/html/2410.22457v1
19. Auction-Based Task Allocation for Multi-robot Teams in Dynamic Environments, accessed November 14, 2025, https://www.researchgate.net/publication/300641858_Auction-Based_Task_Allocation_for_Multi-robot_Teams_in_Dynamic_Environments
20. Auction-Based Behavior Tree Evolution for Heterogeneous Multi-Agent Systems - MDPI, accessed November 14, 2025, https://www.mdpi.com/2076-3417/14/17/7896
21. [2304.01976] Reactive Multi-agent Coordination using Auction-based Task Allocation and Behavior Trees - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2304.01976
22. Intelligent Agents for Auction-based Federated Learning: A ... - IJCAI, accessed November 14, 2025, https://www.ijcai.org/proceedings/2024/0912.pdf
23. How do AI agents handle conflicting objectives? - Milvus, accessed November 14, 2025, https://milvus.io/ai-quick-reference/how-do-ai-agents-handle-conflicting-objectives
24. Pareto Multi-Objective Alignment for Language Models - arXiv, accessed November 14, 2025, https://arxiv.org/html/2508.07768v1
25. Distributional Pareto-Optimal Multi-Objective Reinforcement Learning, accessed November 14, 2025, https://caixq1996.github.io/files/DPMORL-neurips23.pdf
26. Pareto Set Learning for Multi-Objective Reinforcement Learning, accessed November 14, 2025, https://ojs.aaai.org/index.php/AAAI/article/view/34068/36223
27. Pareto Set Learning for Multi-Objective Reinforcement Learning - arXiv, accessed November 14, 2025, https://arxiv.org/html/2501.06773v1
28. Hexaly, CP Optimizer, OR-Tools, Gurobi, Cplex on very large-scale instances of the Job Shop Scheduling Problem (JSSP), accessed November 14, 2025, https://www.hexaly.com/benchmarks/hexaly-vs-cp-optimizer-vs-or-tools-on-the-job-shop-scheduling-problem-jssp
29. Multilevel Constrained Bandits: A Hierarchical Upper Confidence Bound

Approach with Safety Guarantees - MDPI, accessed November 14, 2025, https://www.mdpi.com/2227-7390/13/1/149
30. Murakkab: Resource-Efficient Agentic Workflow Orchestration in Cloud Platforms - arXiv, accessed November 14, 2025, https://arxiv.org/html/2508.18298v1
31. GATES: Cost-aware Dynamic Workflow Scheduling via Graph Attention Networks and Evolution Strategy - arXiv, accessed November 14, 2025, https://arxiv.org/html/2505.12355v2
32. [2510.00685] Stochastic Self-Organization in Multi-Agent Systems - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2510.00685
33. Optimal-Agent-Selection: State-Aware Routing Framework for Efficient Multi-Agent Collaboration - arXiv, accessed November 14, 2025, https://arxiv.org/html/2511.02200v1
34. State-Aware Routing Framework for Efficient Multi-Agent ... - arXiv, accessed November 14, 2025, https://arxiv.org/pdf/2511.02200
35. [Literature Review] Optimal-Agent-Selection: State-Aware Routing Framework for Efficient Multi-Agent Collaboration - Moonlight, accessed November 14, 2025, https://www.themoonlight.io/en/review/optimal-agent-selection-state-aware-routing-framework-for-efficient-multi-agent-collaboration
36. [论文审查] Optimal-Agent-Selection: State-Aware Routing Framework for Efficient Multi-Agent Collaboration - Moonlight, accessed November 14, 2025, https://www.themoonlight.io/zh/review/optimal-agent-selection-state-aware-routing-framework-for-efficient-multi-agent-collaboration
37. Main track accepted papers (Montreal) - IJCAI 2025, accessed November 14, 2025, https://2025.ijcai.org/montreal-main-track-accepted-papers/
38. Adaptive routing protocols for determining optimal paths in AI multi-agent systems: a priority - arXiv, accessed November 14, 2025, https://www.arxiv.org/pdf/2503.07686
39. [2503.07686] Adaptive routing protocols for determining optimal paths in AI multi-agent systems: a priority- and learning-enhanced approach - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2503.07686
40. Learning to Solve Job Shop Scheduling under Uncertainty - arXiv, accessed November 14, 2025, https://arxiv.org/html/2404.01308v1
41. Argo vs Airflow vs Prefect: How Are They Different - Neptune.ai, accessed November 14, 2025, https://neptune.ai/blog/argo-vs-airflow-vs-prefect-differences
42. Prefect vs. Airflow: Which Workflow Orchestrator Should Data Engineers Choose? - Medium, accessed November 14, 2025, https://medium.com/@baheldeepti/prefect-vs-airflow-which-workflow-orchestrator-should-data-engineers-choose-548c8ff6d042
43. Airflow vs Prefect: Deciding Which is Right For Your Data Workflow | DataCamp, accessed November 14, 2025, https://www.datacamp.com/tutorial/airflow-vs-prefect-deciding-which-is-right-for-your-data-workflow
44. Orchestration Showdown: Dagster vs Prefect vs Airflow - ZenML Blog, accessed November 14, 2025,

https://www.zenml.io/blog/orchestration-showdown-dagster-vs-prefect-vs-airflow
45. Agentic Workflows and Prompt Optimization - Predli, accessed November 14, 2025, https://www.predli.com/post/agentic-workflows-and-prompt-optimization
46. Agentic AI Pitfalls: Loops, Hallucinations, Ethical Failures & Fixes | by Amit Kharche, accessed November 14, 2025, https://medium.com/@amitkharche14/agentic-ai-pitfalls-loops-hallucinations-ethical-failures-fixes-77bd97805f9f
47. Multi-Agent Workflows: A Practical Guide to Design, Tools, and Deployment - Kanerika, accessed November 14, 2025, https://kanerika.com/blogs/multi-agent-workflows/
48. MAPS: Multi-Agent Personality Shaping for Collaborative Reasoning - arXiv, accessed November 14, 2025, https://arxiv.org/html/2503.16905v2
49. LLM Agents at the Roundtable: A Multi-Perspective and Dialectical Reasoning Framework for Essay Scoring - arXiv, accessed November 14, 2025, https://arxiv.org/html/2509.14834v1
50. Multi-Model Dialectical Evaluation of LLM Reasoning Chains: A Structured Framework with Dual Scoring Agents - MDPI, accessed November 14, 2025, https://www.mdpi.com/2227-9709/12/3/76
51. # The Dialectical Agent: A Speculative Framework for Emergent Meta-Reasoning Through Internal... | by Xenonostra' | Sep, 2025 | Medium, accessed November 14, 2025, https://medium.com/@tmcrg85/the-dialectical-agent-a-speculative-framework-for-emergent-meta-reasoning-through-internal-480704f159a1
52. AI Agentic Programming: A Survey of Techniques, Challenges, and Opportunities - arXiv, accessed November 14, 2025, https://arxiv.org/html/2508.11126v1
53. How to Define Success in Multi-Agent AI Systems | Galileo, accessed November 14, 2025, https://galileo.ai/blog/success-multi-agent-ai
54. A Detailed Comparison of Top 6 AI Agent Frameworks in 2025 - Turing, accessed November 14, 2025, https://www.turing.com/resources/ai-agent-frameworks
55. Autogen vs CrewAI vs LangGraph 2025 Comparison Guide - Python in Plain English, accessed November 14, 2025, https://python.plainenglish.io/autogen-vs-crewai-vs-langgraph-2025-comparison-guide-7cad22747f11
56. A Study on Multi-Arm Bandit Problem with UCB and Thompson Sampling Algorithm - SciTePress, accessed November 14, 2025, https://www.scitepress.org/Papers/2024/129384/129384.pdf
57. Constrained Feedback Learning for Non-Stationary Multi-Armed Bandits - arXiv, accessed November 14, 2025, https://arxiv.org/pdf/2509.15073
58. Adapting to Non-Stationary Environments: Multi-Armed Bandit Enhanced Retrieval-Augmented Generation on Knowledge Graphs - arXiv, accessed November 14, 2025, https://arxiv.org/html/2412.07618v1
59. Multi-Armed Bandits Meet Large Language Models - arXiv, accessed November 14, 2025, https://arxiv.org/html/2505.13355v1
60. Multi Armed Bandit Algorithms Based Virtual Machine Allocation Policy for

Security in Multi-Tenant Distributed Systems. - arXiv, accessed November 14, 2025, https://arxiv.org/html/2410.04363v1
61. Thompson Sampling for Constrained Bandits - Reinforcement ..., accessed November 14, 2025, https://rlj.cs.umass.edu/2025/papers/RLJ_RLC_2025_365.pdf
62. Thompson Sampling for Constrained Bandits - OpenReview, accessed November 14, 2025, https://openreview.net/forum?id=5HL4QaNQcf
63. arXiv:2405.11417v1 [cs.LG] 19 May 2024, accessed November 14, 2025, https://arxiv.org/pdf/2405.11417?
64. Adaptive Budgeted Multi-Armed Bandits for IoT with Dynamic Resource Constraints - arXiv, accessed November 14, 2025, https://arxiv.org/pdf/2505.02640
65. Bi-Level Contextual Bandits for Individualized Resource Allocation under Delayed Feedback† - arXiv, accessed November 14, 2025, https://arxiv.org/html/2511.10572v1
66. [2511.10572] Bi-Level Contextual Bandits for Individualized Resource Allocation under Delayed Feedback - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2511.10572
67. An experimental survey and Perspective View on Meta-Learning for Automated Algorithms Selection and Parametrization - arXiv, accessed November 14, 2025, https://arxiv.org/pdf/2504.06207
68. [2411.00625] Toward Automated Algorithm Design: A Survey and Practical Guide to Meta-Black-Box-Optimization - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2411.00625
69. Toward Automated Algorithm Design: A Survey and Practical Guide to Meta-Black-Box-Optimization - arXiv, accessed November 14, 2025, https://arxiv.org/html/2411.00625v1
70. Cost-Efficient Training for Automated Algorithm Selection - OpenReview, accessed November 14, 2025, https://openreview.net/forum?id=BJEQvOAxrP
71. Opening the Black Box: Automated Software Analysis for Algorithm Selection | OpenReview, accessed November 14, 2025, https://openreview.net/forum?id=Sgg5KAwNg9
72. Toward Automated Algorithm Design: A Survey and Practical Guide to Meta-Black-Box-Optimization - arXiv, accessed November 14, 2025, https://arxiv.org/html/2411.00625v3
73. MetaBox: A Benchmark Platform for Meta-Black-Box Optimization with Reinforcement Learning | OpenReview, accessed November 14, 2025, https://openreview.net/forum?id=j2wasUypqN¬eId=xnSCKeUXBs
74. Population-Based Training (PBT) Hyperparameter Tuning - DZone, accessed November 14, 2025, https://dzone.com/articles/population-based-training-pbt-hyperparameter-tunin
75. Multiple-Frequencies Population-Based Training - arXiv, accessed November 14, 2025, https://arxiv.org/html/2506.03225v1
76. A Guide to Population Based Training with Tune - Ray Docs, accessed November 14, 2025, https://docs.ray.io/en/latest/tune/examples/pbt_guide.html
77. Multi-Objective Population Based Training - Proceedings of Machine Learning

Research, accessed November 14, 2025, https://proceedings.mlr.press/v202/dushatskiy23a/dushatskiy23a.pdf
78. Generalized Population-Based Training for Hyperparameter Optimization in Reinforcement Learning - arXiv, accessed November 14, 2025, https://arxiv.org/html/2404.08233v1
79. Large Language Model Evaluation: 10+ Metrics & Methods - Research AIMultiple, accessed November 14, 2025, https://research.aimultiple.com/large-language-model-evaluation/
80. am-ELO: A Stable Framework for Arena-based LLM Evaluation - arXiv, accessed November 14, 2025, https://arxiv.org/html/2505.03475v1
81. Chatbot Arena: Benchmarking LLMs in the Wild with Elo Ratings | LMSYS Org, accessed November 14, 2025, https://lmsys.org/blog/2023-05-03-arena/
82. Chatbot Arena and the Elo rating system - Part 1 - Yi Zhu, accessed November 14, 2025, https://bryanyzhu.github.io/posts/2024-06-20-elo-part1/
83. Elo as a tool for ranking LLMs. In a previous blog post, Vikas and... | by Rahul | Thomson Reuters Labs | Medium, accessed November 14, 2025, https://medium.com/tr-labs-ml-engineering-blog/elo-as-a-tool-for-ranking-llms-dab056dc9713
84. EvoFlow: Evolving Diverse Agentic Workflows On The Fly - arXiv, accessed November 14, 2025, https://arxiv.org/pdf/2502.07373?
85. EvoFlow: Evolving Diverse Agentic Workflows On The Fly | Request PDF - ResearchGate, accessed November 14, 2025, https://www.researchgate.net/publication/388920791_EvoFlow_Evolving_Diverse_Agentic_Workflows_On_The_Fly
86. EvoAgentX: An Automated Framework for Evolving Agentic Workflows - arXiv, accessed November 14, 2025, https://arxiv.org/html/2507.03616v2
87. EvoAgentX: Building a Self-Evolving Ecosystem of AI Agents - GitHub, accessed November 14, 2025, https://github.com/EvoAgentX/EvoAgentX
88. EvoAgentX: An Automated Framework for Evolving Agentic ..., accessed November 14, 2025, https://aclanthology.org/2025.emnlp-demos.47/
89. EvoAgentX: An Automated Framework for Evolving Agentic Workflows - ChatPaper, accessed November 14, 2025, https://chatpaper.com/paper/158847
90. EvoAgentX: An Automated Framework for Evolving Agentic Workflows - ACL Anthology, accessed November 14, 2025, https://aclanthology.org/2025.emnlp-demos.47.pdf
91. AFlow: Automating Agentic Workflow Generation - OpenReview, accessed November 14, 2025, https://openreview.net/forum?id=z5uVAKwmjf
92. EvoAgentX: An Automated Framework for Evolving Agentic Workflows | Request PDF, accessed November 14, 2025, https://www.researchgate.net/publication/397422261_EvoAgentX_An_Automated_Framework_for_Evolving_Agentic_Workflows
93. [2507.03616] EvoAgentX: An Automated Framework for Evolving Agentic Workflows - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2507.03616
94. EVOAGENT: Towards Automatic Multi-Agent Generation via Evolutionary Algorithms - ACL Anthology, accessed November 14, 2025,

https://aclanthology.org/2025.naacl-long.315.pdf
95. [2502.02533] Multi-Agent Design: Optimizing Agents with Better Prompts and Topologies, accessed November 14, 2025, https://arxiv.org/abs/2502.02533
96. Revision History for Multi-Agent Design: Optimizing Agents... - OpenReview, accessed November 14, 2025, https://openreview.net/revisions?id=V4kLWUSwsw
97. Multi-Agent Design: Optimizing Agents with Better Prompts and Topologies, accessed November 14, 2025, https://research.google/pubs/multi-agent-design-optimizing-agents-with-better-prompts-and-topologies/
98. Multi-Agent Design: Optimizing Agents with Better Prompts and Topologies - Hugging Face, accessed November 14, 2025, https://huggingface.co/papers/2502.02533
99. Multi-Agent Design: Optimizing Agents with Better Prompts and Topologies - arXiv, accessed November 14, 2025, https://arxiv.org/html/2502.02533v1
100. NADER: Neural Architecture Design via Multi-Agent Collaboration - arXiv, accessed November 14, 2025, https://arxiv.org/html/2412.19206v1
101. NADER: Neural Architecture Design via Multi-Agent Collaboration - Zekang Yang, accessed November 14, 2025, https://yang-ze-kang.github.io/NADER/
102. Official implementation for "NADER: Neural Architecture Design via Multi-Agent Collaboration" (CVPR2025) - GitHub, accessed November 14, 2025, https://github.com/yang-ze-kang/NADER
103. [2412.19206] NADER: Neural Architecture Design via Multi-Agent Collaboration - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2412.19206
104. NADER: Neural Architecture Design via Multi-Agent Collaboration - CVF Open Access, accessed November 14, 2025, https://openaccess.thecvf.com/content/CVPR2025/papers/Yang_NADER_Neural_Architecture_Design_via_Multi-Agent_Collaboration_CVPR_2025_paper.pdf
105. DEAP/deap: Distributed Evolutionary Algorithms in Python - GitHub, accessed November 14, 2025, https://github.com/DEAP/deap
106. arXiv:2410.21940v1 [cs.LG] 29 Oct 2024, accessed November 14, 2025, https://www.arxiv.org/pdf/2410.21940
107. arXiv:2406.18532v1 [cs.CL] 26 Jun 2024, accessed November 14, 2025, https://arxiv.org/pdf/2406.18532
108. Scalable Constrained Policy Optimization for Safe Multi-agent Reinforcement Learning - NIPS papers, accessed November 14, 2025, https://proceedings.neurips.cc/paper_files/paper/2024/file/fa76985f05e0a25c66528308dda33de0-Paper-Conference.pdf
109. A Comprehensive Survey on Multi-Agent Cooperative Decision-Making: Scenarios, Approaches, Challenges and Perspectives - arXiv, accessed November 14, 2025, https://arxiv.org/html/2503.13415v1
110. RobustFlow: Towards Robust Agentic Workflow Generation - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2509.21834
111. [2211.16072] A Brief Introduction to Robust Bilevel Optimization - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2211.16072
112. OSU-NLP-Group/AgentSafety - GitHub, accessed November 14, 2025,

https://github.com/OSU-NLP-Group/AgentSafety
113. DeepTeam is a framework to red team LLMs and LLM systems. - GitHub, accessed November 14, 2025, https://github.com/confident-ai/deepteam
114. SnailSploit/Aversarial-AI-Threat-Modeling-Framwork: AATMF | An Open Source - GitHub, accessed November 14, 2025, https://github.com/SnailSploit/Aversarial-AI-Threat-Modeling-Framwork
115. [2307.16212] Robust Multi-Agent Reinforcement Learning with State Uncertainty - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2307.16212
116. [2312.10256] Multi-agent Reinforcement Learning: A Comprehensive Survey - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2312.10256
117. Multi-agent Reinforcement Learning: A Comprehensive Survey - arXiv, accessed November 14, 2025, https://arxiv.org/html/2312.10256v2
118. Sample-Efficient Robust Multi-Agent Reinforcement Learning in the Face of Environmental Uncertainty - arXiv, accessed November 14, 2025, https://arxiv.org/html/2404.18909v3
119. A Game-Theoretic Approach to Multi-Agent Interaction for Increased Coordination and Safety in Robotics - UPCommons, accessed November 14, 2025, https://upcommons.upc.edu/bitstreams/298a44ea-4e8e-4734-9c51-dd542bbd5573/download
120. Stackelberg Game-Theoretic Learning for Collaborative Assembly Task Planning - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2404.12570
121. Stackelberg Game-Theoretic Learning for Collaborative Assembly Task Planning - arXiv, accessed November 14, 2025, https://arxiv.org/html/2404.12570v1
122. [2507.02002] Dynamic Strategy Adaptation in Multi-Agent Environments with Large Language Models - arXiv, accessed November 14, 2025, https://arxiv.org/abs/2507.02002
123. My thoughts on the most popular frameworks today: crewAI, AutoGen, LangGraph, and OpenAI Swarm : r/LangChain - Reddit, accessed November 14, 2025, https://www.reddit.com/r/LangChain/comments/1g6i7cj/my_thoughts_on_the_most_popular_frameworks_today/
124. Choosing the Right AI Agent Framework: LangGraph vs CrewAI vs OpenAI Swarm - nuvi, accessed November 14, 2025, https://www.nuvi.dev/blog/ai-agent-framework-comparison-langgraph-crewai-openai-swarm
125. Exploring the Future of Agentic AI Swarms - Codewave, accessed November 14, 2025, https://codewave.com/insights/future-agentic-ai-swarms/
126. LLM-based Multi-Agent Blackboard System for Information Discovery in Data Science - arXiv, accessed November 14, 2025, https://arxiv.org/html/2510.01285v1
127. Building Multi-Agent Architectures ⟶ Orchestrating Intelligent Agent Systems | by Akanksha Sinha | Medium, accessed November 14, 2025, https://medium.com/@akankshasinha247/building-multi-agent-architectures-orchestrating-intelligent-agent-systems-46700e50250b

128. Exploring Advanced LLM Multi-Agent Systems Based on Blackboard Architecture, accessed November 14, 2025, https://www.researchgate.net/publication/393333734_Exploring_Advanced_LLM_Multi-Agent_Systems_Based_on_Blackboard_Architecture
129. Exploring Advanced LLM Multi-Agent Systems Based on Blackboard Architecture - arXiv, accessed November 14, 2025, https://arxiv.org/html/2507.01701v1
130. SwarmSys: Decentralized Swarm-Inspired Agents for Scalable and Adaptive Reasoning, accessed November 14, 2025, https://arxiv.org/html/2510.10047v1
131. SOCIALJAX: AN EVALUATION SUITE FOR MULTI ... - OpenReview, accessed November 14, 2025, https://openreview.net/pdf/673d42b5eb9b37162e598877bb6a7ea5fc7e4cc8.pdf
132. Examples — Ray 2.51.1 - Ray Docs, accessed November 14, 2025, https://docs.ray.io/en/latest/rllib/rllib-examples.html
133. [RLlib] adapting self_play_with_open_spiel to work with pettingzoo.classic environments · Issue #32382 · ray-project/ray - GitHub, accessed November 14, 2025, https://github.com/ray-project/ray/issues/32382
134. Help With RLLib/ Alternatives : r/reinforcementlearning - Reddit, accessed November 14, 2025, https://www.reddit.com/r/reinforcementlearning/comments/165ivu8/help_with_rllib_alternatives/
135. Deep Reinforcement Learning: A Practical Guide for Game Devs - Medium, accessed November 14, 2025, https://medium.com/@sentiencegamestudio/deep-reinforcement-learning-a-practical-guide-for-game-devs-93541badc4ca
136. The Ultimate Guide to Migrating from Gurobi to CPLEX | Cresco International, accessed November 14, 2025, https://crescointl.com/the-ultimate-guide-to-migrating-from-gurobi-to-cplex/
137. Mastering Hyperparameter Optimization: The Best Frameworks & Code Examples - Medium, accessed November 14, 2025, https://medium.com/@wijdnbouzidii/mastering-hyperparameter-optimization-the-best-frameworks-code-examples-781fc92e959e
138. 10 Hyperparameter optimization frameworks - Kaggle, accessed November 14, 2025, https://www.kaggle.com/code/sivasaiyadav8143/10-hyperparameter-optimization-frameworks
139. [D] Hyperparameter Optimization with Evolutionary Algorithms: A Biological Approach to Adaptive Search : r/MachineLearning - Reddit, accessed November 14, 2025, https://www.reddit.com/r/MachineLearning/comments/1lqw0fj/d_hyperparameter_optimization_with_evolutionary/
140. DEAP: Evolutionary Algorithms Made Easy - Journal of Machine Learning Research, accessed November 14, 2025, https://www.jmlr.org/papers/volume13/fortin12a/fortin12a.pdf
141. 10 AI agent benchmarks - Evidently AI, accessed November 14, 2025,

https://www.evidentlyai.com/blog/ai-agent-benchmarks
142. Evaluation and Benchmarking of LLM Agents: A Survey - arXiv, accessed November 14, 2025, https://arxiv.org/html/2507.21504v1
143. How we built our multi-agent research system - Anthropic, accessed November 14, 2025, https://www.anthropic.com/engineering/multi-agent-research-system
144. Top 4 Agentic AI Design Patterns for Architecting AI Systems - Analytics Vidhya, accessed November 14, 2025, https://www.analyticsvidhya.com/blog/2024/10/agentic-design-patterns/
145. Orchestrating Human-AI Teams: The Manager Agent as a Unifying Research Challenge, accessed November 14, 2025, https://arxiv.org/html/2510.02557v1
146. Hierarchical Frameworks for Scaling-up Multi-agent Coordination - IFAAMAS, accessed November 14, 2025, https://www.ifaamas.org/Proceedings/aamas2025/pdfs/p2932.pdf
147. Multi-agent Orchestration deep dive - collaboration patterns from MetaGPT to AutoGen : r/pythontips - Reddit, accessed November 14, 2025, https://www.reddit.com/r/pythontips/comments/1ntn1xp/multiagent_orchestration_deep_dive_collaboration/
148. Stigmergy | Swarm Intelligence and Robotics Class Notes | Fiveable, accessed November 14, 2025, https://fiveable.me/swarm-intelligence-and-robotics/unit-6/stigmergy/study-guide/L6j1cyesyCpC1JCs
149. SwarmSys: Decentralized Swarm-Inspired Agents for Scalable and Adaptive Reasoning, accessed November 14, 2025, https://www.researchgate.net/publication/396458001_SwarmSys_Decentralized_Swarm-Inspired_Agents_for_Scalable_and_Adaptive_Reasoning
150. tmgthb/Autonomous-Agents: Autonomous Agents (LLMs) research papers. Updated Daily. - GitHub, accessed November 14, 2025, https://github.com/tmgthb/Autonomous-Agents
151. Daily Papers - Hugging Face, accessed November 14, 2025, https://huggingface.co/papers?q=long-horizon%20reasoning
152. A Concept for Bio-Agentic Visual Communication: Bridging Swarm Intelligence with Biological Analogues - PMC - PubMed Central, accessed November 14, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC12467162/