On-Demand Multi-Agent Workflow Synthesis for Multimodal Long-Horizon Reasoning

Dmitry Gilemkhanov [*1] **Jerzy Kamiński** [*1] **Vadim Akhmerov** [1] **Ivan Dubrovsky** [1] **Vadim A. Potemkin** [1] **Alina Zhidkovskaya** [1] **Maria Zaitseva** [1] **Artur Khairullin** [1] **Nikita Kulin** [1] **Sergey Muravyov** [1] **Ilia Revin** [1] **Anna Kalyuzhnaya** [1] **Ravil Mussabayev** [2] **Alexander Boukhanovsky** [1] **Andrey Savchenko** [3 4] **Nikolay O. Nikitin** [1]

Abstract

Multimodal agentic tasks—spanning web operations, software workflows, and tool-augmented reasoning over heterogeneous information sources—increasingly require coordinating multiple specialized agents rather than scaling a single generalist through longer reasoning chains. Designing such multi-agent systems remains combinatorially hard: existing approaches commit to fixed templates and domain-specific training, or deploy autonomous generalists at growing inference cost. We introduce **TSAS** (Two-Stage Agent Synthesis), a framework that generates task-specific multi-agent systems on demand. Given a task description, a single meta-agent call produces both a pool of specialized agents and a directed acyclic graph defining their coordination, without iterative search or additional training. On the GAIA benchmark, which probes long-horizon reasoning across web search, image, audio, and structured-file modalities, TSAS achieves 47.18% accuracy with GPT-5-mini at $31.58 per run, matching the top autonomous system while reducing cost by 61%, and outperforming agents built on Claude Opus 4 and Claude-4.5 Sonnet by over 16 percentage points. We further show that workflow generation can be distilled: a fine-tuned meta-agent (GraphGen-14B, based on Qwen2.5-Coder) reaches 57.57% on GAIA validation, more than doubling prior fine-tuned baselines.

[1]ITMO University, St. Petersburg, Russia [2]Satbayev University, Almaty, Kazakhstan [3]Sber AI Lab, Moscow, Russia [4]Higher School of Economics, Moscow, Russia. Correspondence to: Dmitry Gilemkhanov <me@dmitryglhf.ru>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

1. Introduction

Multimodal agentic systems—agents that coordinate web search, document understanding, code execution, and tool invocation to solve long-horizon tasks—have emerged as a central paradigm for problems that single forward passes cannot solve (Mialon et al., 2023). Real-world deployments span software engineering, web and mobile operations, embodied manipulation, and scientific workflows, where progress depends on integrating heterogeneous modalities—images, audio, structured files, web content—within structured reasoning trajectories. As these settings scale, two paradigms have emerged for organizing work across multiple components, each with characteristic limitations.

The first paradigm composes specialized LLM-based agents into explicit workflows (Wu et al., 2023; Hong et al., 2024; Li et al., 2023), but workflow design is combinatorially hard. Existing automation approaches—template modification (Zhang et al., 2024a; Wang et al., 2025c; Zhang et al., 2024b), learning from task-workflow pairs (Ye et al., 2025; Li et al., 2025a), and search-based refinement (Zhang et al., 2025a; 2024c; Wang et al., 2025b)—all attempt to discover effective workflows through offline optimization, leaving them ill-suited to the novel configurations that complex multimodal tasks demand. MAS-GPT (Ye et al., 2025), for instance, achieves only 23.03% on GAIA despite training a larger model, and our own experiments confirm that post-hoc search yields negligible gains at 6–9× the inference cost.

The second paradigm sidesteps workflow design entirely, deploying autonomous generalist agents that scale performance through extended reasoning chains and test-time compute. While capable, these systems trade structural transparency for opaque end-to-end reasoning at costs exceeding $270–450 per evaluation (Table 1), and their gains plateau without architectural inductive bias matched to the task at hand.

We propose **TSAS** (Two-Stage Agent Synthesis), a framework that generates task-specific workflows on demand

through meta-agent reasoning. Given a task description, TSAS employs a two-stage generation process: (1) synthesizing a pool of agents with distinct capabilities and tool access, and (2) constructing a directed acyclic graph that defines agent dependencies and information flow. This decomposition mirrors the distinction between capability definition and coordination protocol in classical multi-agent systems (Wooldridge, 2009), enabling the meta-agent to reason separately about *what* agents are needed and *how* they should interact.

Our approach offers three key advantages:

1. Dynamic adaptation. Unlike template-based approaches constrained by fixed topologies, TSAS generates workflows tailored to each task's requirements, producing single-agent solutions for simple queries and multi-agent collaborations for complex multimodal problems.

2. Zero-shot generalization. By leveraging LLM reasoning rather than learned task-topology mappings, TSAS handles novel task types without domain-specific training data or architectural modifications. The meta-agent works out of the box with any instruction-following LLM.

3. Distillation to smaller models. While training-free by default, workflow generation can optionally be transferred to compact models through supervised fine-tuning (SFT), yielding **GraphGen-14B**, a model that generates workflows at a fraction of the inference cost.

We evaluate TSAS on the GAIA benchmark, which probes multimodal long-horizon reasoning across 165 validation and 301 test tasks spanning web search, image and audio understanding, structured file processing, and tool-augmented computation (Mialon et al., 2023). Our main findings are:

- On GAIA, TSAS achieves 47.18% at $31.58 per run, matching top autonomous systems while cutting costs by 61% (Table 1).
- Fine-tuned GraphGen-14B achieves 57.57% on GAIA validation, more than doubling the MAS-GPT baseline (23.03%) and outperforming RL-based training (31.52%), showing that SFT on demonstrations suffices for learning workflow generation (Table 2).

2. Related Work

Template-based and graph-modification approaches. A prominent line of work modifies predefined template graphs to obtain task-specific topologies. AgentPrune (Zhang et al., 2024a) learns sparse topologies via edge pruning; AgentDropout (Wang et al., 2025c) extends this with stochastic dropout at execution time; G-Designer (Zhang et al., 2024b) employs a GNN-based autoencoder for richer topology transformations. All three share the same constraint: the agent pool is predetermined and the topology space is bounded by the initial structure.

Autoregressive and learning-based generation. ARG-Designer (Li et al., 2025a) reframes topology design as autoregressive graph generation, building graphs from scratch through sequential agent selection and edge construction with metric-learning-based role selection. MAS-GPT (Ye et al., 2025) fine-tunes an LLM directly on task-workflow pairs. These methods rely on synthetic training data with success signals; for each new task domain, additional supervision is required.

Optimization-based and broader automated design. Beyond topology editing, automated MAS design spans GNN- and RL-based optimization (ADAS (Hu et al., 2025), GPTSwarm (Zhuge et al., 2024), AgentSquare (Shang et al., 2024), AgentSwift (Li et al., 2025b), Multi-agent Architecture Search (Zhang et al., 2025b)), sequential protocols for dynamic agent activation (AnyMAC (Wang et al., 2025a), Flow (Niu et al., 2025), Debflow (Su et al., 2025)), evolutionary refinement (EvoFlow (Zhang et al., 2025a), EvoAgent (Yuan et al., 2024), ScoreFlow (Wang et al., 2025b)), and template-based generation (Autoflow (Li et al., 2024), AFlow (Zhang et al., 2024c)). All require either iterative refinement or labeled task-workflow data.

Multimodal and embodied agents. A complementary line of work targets perception-grounded reasoning—vision-language agents, embodied agents for manipulation and navigation, and multimodal foundation models orchestrating tool use. These efforts are orthogonal to the meta-level question of workflow composition; TSAS-generated DAGs are agnostic to the modality each agent processes through its assigned tools.

Our approach. We propose on-demand workflow generation: a meta-agent designs task-specific workflows at inference time by leveraging the LLM's reasoning capability rather than learned mappings or iterative search. The result is a minimal workflow for simple queries and an elaborate collaboration for complex multimodal problems, generated in a single forward pass without training data. We additionally fine-tune compact models (Qwen2.5-Coder-14B (Hui et al., 2024), Llama-3-8B (AI@Meta, 2024)) to internalize this capability at lower inference cost.

3. Method

3.1. Problem Formulation

We consider the problem of automatically constructing and executing multi-agent systems for solving complex tasks. Let $\mathcal{T}$ denote a task specified in natural language, and let $\mathcal{D} = \{d_1, \ldots, d_k\}$ be a set of available tool descriptions. Our goal is to learn a mapping from task specifications to

executable multi-agent workflows that produce a solution r.

We decompose this into three stages: (1) *agent synthesis*: a pool of specialized agents $\mathcal{A}$ is generated tailored to the task; (2) *workflow design*: a directed acyclic graph $\mathcal{G}$ is constructed defining agent dependencies; (3) *coordinated execution*: agents are executed respecting dependency constraints.

3.2. System Definition

An *agent* $a \in \mathcal{A}$ is defined as a tuple $a = (\pi, m, \mathcal{T}_a)$, where π is the instruction prompt, $m \in \mathcal{M}$ is the language model, and $\mathcal{T}_a \subseteq \mathcal{D}$ is the tool subset. A *workflow graph* $\mathcal{G} = (\mathcal{A}, E)$ is a DAG over agents, where an edge $(a_i, a_j) \in E$ indicates that the output of a_i is provided as input to a_j.

We employ two *meta-agents* parameterized by LLMs: a *pool generator* M_p that maps a task description to a set of specialized agents, and a *graph generator* M_g that constructs a workflow DAG over the agent pool.

For execution, we assign each agent to a *level* $\ell(a)$ based on the longest path from any entry node:

$$\ell(a) = \begin{cases} 0, & \text{if } a \text{ has no parents} \\ 1 + \max_{p \in \mathsf{parents}(a)} \ell(p), & \text{otherwise} \end{cases} \tag{1}$$

Agents at the same level have no mutual dependencies and execute in parallel. Each agent receives a context aggregated from its parents' outputs and produces a result propagated to its children.

3.3. Agent Pool Synthesis

Given a task $\mathcal{T}$ and tool descriptions $\mathcal{D}$, the pool generator M_p produces a set of specialized agents $\mathcal{A} = M_p(\mathcal{T}, \mathcal{D})$. The meta-agent analyzes the task requirements and determines the number of agents adaptively: single-agent configurations for straightforward queries, and pools of 2–5 agents for tasks requiring distinct capabilities or parallel information gathering across modalities.

For each agent $a_i \in \mathcal{A}$, the generator specifies: (1) a unique descriptive name, (2) an instruction prompt π_i defining the agent's role and expected behavior, (3) a model assignment $m_i \in \mathcal{M}$, and (4) a tool subset $\mathcal{T}_{a_i} \subseteq \mathcal{D}$ selected from the available MCP server registry. The generator validates that all agent names are unique and that every assigned tool exists in the registry, rejecting malformed configurations.

3.4. Workflow Graph Construction

Given the task $\mathcal{T}$ and agent pool $\mathcal{A}$, the graph generator M_g constructs a DAG $\mathcal{G} = (\mathcal{A}, E) = M_g(\mathcal{T}, \mathcal{A})$. The output is an adjacency list mapping each agent to its dependents, where an edge (a_i, a_j) indicates that a_j requires a_i's output. Acyclicity is enforced by construction: the meta-agent generates only forward-directed edges, and the resulting structure is verified via topological sort before execution.

Separating pool synthesis from graph construction is a deliberate design choice. The ablation in Section 4.5 confirms that this two-stage decomposition outperforms unified generation by 3 percentage points, as it allows the graph generator to reason about inter-agent dependencies with full knowledge of each agent's capabilities.

3.5. Pipeline Construction and Execution

Algorithm 1 presents the full procedure. The pool generator M_p synthesizes specialized agents $\mathcal{A}$, then M_g produces a DAG $\mathcal{G}$. We apply topological sorting and group nodes into levels $\mathcal{L}$. Execution proceeds level-by-level, with parallel execution within each level, and outputs are cached in $\mathcal{E}$. The context aggregation function ϕ concatenates the task description q with the outputs of all parent agents into a structured prompt, providing each agent with the original task and all upstream results.

Algorithm 1 Multi-Agent System Construction and Execution

Require: Task $\mathcal{T}$, tool descriptions $\mathcal{D}$, meta-agents M_p, M_g
Ensure: Result r

```
A ← M_p(T, D)
G = (A, E) ← M_g(T, A)
order ← TopologicalSort(G)
L ← GroupByLevel(order)
E ← ∅
for i = 0, ..., |L| − 1 do
    for each a ∈ L_i in parallel do
        c_a ← φ(q, {o_p : p ∈ parents(a)})
        E[a] ← Execute(a, c_a)
    end for
end for
return E[order_{−1}]
```

3.6. Optional Prompt Refinement

After DAG construction, an optional prompt refinement step can be applied using the Hypothetical Prompt Enhancer (HyPE) (Kulin et al., 2025), a zero-shot, single-step meta-optimization method that leverages the LLM's internal knowledge to produce more structured instruction prompts. We adapt HyPE with role-aware refinement based on each agent's topological position: terminal agents receive prompts emphasizing strict output-format adherence, while intermediate agents receive prompts focused on outputs immediately actionable by downstream agents. The refinement adds one LLM call per agent and is evaluated in Section 4.8.

3.7. Walkthrough: A Multimodal GAIA Task

To make the framework concrete, consider a representative GAIA Level 3 task: *"Which of the fruits shown in the 2008 painting 'Embroidery from Uzbekistan' were served as part of the October 1949 breakfast menu for the ocean liner that was later used as a floating prop for the film 'The Last Voyage'? Give the items as a comma-separated list, ordering them in clockwise order according to their arrangement in the painting, starting from the 12 o'clock position."* The task spans three modalities: visual analysis of the painting, web research over film and shipping history, and retrieval of an archival textual menu, with a strict output format applied at the end.

The pool generator M_p analyzes the task and synthesizes three specialized agents:

- **ArtResearchAgent**—locates the painting and extracts the depicted fruits and their spatial arrangement. Tools: `web-search`, `web-extract`, `vision`.
- **ShipHistoryAgent**—identifies the ocean liner from film production records and retrieves the October 1949 breakfast menu. Tools: `web-search`, `web-extract`.
- **AnalysisAgent**—cross-references findings and applies the formatting constraints. No external tools.

The graph generator M_g then constructs a two-level DAG: `ArtResearchAgent` and `ShipHistoryAgent` occupy level 0 and execute in parallel, both feeding into `AnalysisAgent` at level 1, which serves as the terminal node. The structure illustrates the meta-agent's adaptive coordination: parallel branches for independent multimodal subtasks, sequential merge at the synthesis stage. Crucially, this configuration is generated by M_p and M_g in a single forward pass per stage from the task description alone, without retrieval over a workflow library or iterative search over candidate topologies.

4. Experimental Study

4.1. Setup

Benchmark. We evaluate primarily on GAIA (Mialon et al., 2023), a multimodal long-horizon agentic benchmark comprising 165 validation and 301 test tasks across three difficulty levels. GAIA tasks require integrating web search, image and audio understanding, structured file processing (PDFs, spreadsheets, code), and tool-augmented computation, making it directly aligned with the multimodal agentic deployment scenarios that motivate this work.

Baselines. For GAIA, we compare against autonomous agents on the public HAL leaderboard[1], including HF Open Deep Research and HAL Generalist configurations on flagship backbones (Claude Opus 4, Claude-4.5 Sonnet, o4-mini, DeepSeek V3, Gemini 2.0 Flash). For automated workflow generation, we compare against MAS-GPT (Ye et al., 2025) and RL-based fine-tuning, both evaluated on GAIA validation.

Implementation Details. For meta-agent generation, we use Claude Sonnet 4 as the default backbone. Generated workflows are executed with tool access including web search, code execution, image and audio understanding, and file manipulation. All experiments use greedy decoding with temperature 0. Cost estimates include both workflow generation and inference phases.

4.2. Main Results on GAIA

Table 1 presents results on the GAIA test set compared to the HAL leaderboard. Our approach uses GPT-5-mini as the workflow executor, a smaller and older model than the flagship systems dominating the leaderboard. Despite this, TSAS achieves 47.18% accuracy at \$31.58, matching the top-performing HF Open Deep Research system (47.88%) while reducing costs by 61%. The same HF system requires Claude-4.5 Sonnet at \$451.84 to reach 30.91%, or o4-mini at \$80.80 to reach 47.88%. TSAS-generated workflows also outperform autonomous agents built on Claude Opus 4 (30.30%, \$272.76) and Claude-4.5 Sonnet (30.91%, \$451.84) by over 16 percentage points, demonstrating that explicit workflow decomposition compensates for weaker base models on multimodal long-horizon tasks.

To isolate the contribution of workflow generation from the choice of executor backbone, we additionally evaluate TSAS on DeepSeek V3, the same model used by HAL Generalist on the leaderboard. Under this controlled comparison, TSAS reaches 36.88% versus 29.39% for HAL Generalist—a 25.5% relative improvement—while the meta-agent generation overhead amounts to just \$6.45 across all 301 tasks (\$23.85 vs \$17.40 total). The gain therefore stems from the explicit two-stage workflow synthesis rather than from differences in the underlying language model.

4.3. Cross-Backbone Analysis

To assess sensitivity to the executor backbone, we run the two-stage configuration on five language models spanning the current frontier of open and proprietary systems (Figure 1). GPT-5-mini achieves the highest overall accuracy (50.3%), with the per-level breakdown remaining roughly proportional across all backbones—accuracy on Level 3 tasks is consistently the bottleneck, indicating that task complexity rather than coordination overhead drives the absolute

[1] https://hal.cs.princeton.edu/gaia

Table 1. Comparison with the HAL leaderboard on the GAIA test set (301 tasks).

METHOD	MODEL	AVG	COST
HF OPEN DR	O4-MINI LOW	47.88	$80.80
HF OPEN DR	CLAUDE-3.7 SONNET	36.97	$415.15
HAL GEN.	GEMINI 2.0 FLASH	32.73	$7.80
HF OPEN DR	CLAUDE-4.5 SONNET	30.91	$451.84
HAL GEN.	CLAUDE OPUS 4	30.30	$272.76
HAL GEN.	DEEPSEEK V3	29.39	$17.40
TSAS (OURS)	GPT-5-MINI	47.18	$31.58
	DEEPSEEK V3	36.88	$23.85

ceiling. The cost panel reveals a non-monotonic relationship between price and performance: GLM-4.6 is the most expensive backbone ($28.8 per run) yet trails Qwen3-235B ($5.8) by 6.7 pp, and DeepSeek-V3 sits at the cost-efficiency corner ($4.2) at the price of substantially reduced accuracy. Workflow synthesis cost is approximately constant across backbones (∼$3.5), so the cost gap is dominated by inference-time tool use and reasoning depth rather than meta-agent overhead. This decomposition supports a key design assumption: the meta-agent's contribution to total cost is bounded and predictable, leaving the deployment-time choice of executor as the primary lever on the cost-accuracy trade-off.

4.4. Distillation to Smaller Models

To validate the hypothesis that workflow generation can be distilled into compact models, we conducted SFT on a specialized meta-agent (Table 2). We curated a dataset by aggregating logs from 15 runs on the GAIA benchmark, yielding 2,432 examples. Each training instance consists of a system prompt (tailored for either pool synthesis or graph construction), a GAIA task, and the target valid JSON output representing the final workflow configuration. Training instances did not contain any flag indicating whether the resulting workflow produced a correct or incorrect answer.

We fine-tuned Qwen2.5-Coder-14B-Instruct using the Unsloth framework (Daniel Han & team, 2023) with LoRA ($r = 128$, $\alpha = 256$), without quantization, effective batch size 8, learning rate 1e-5, and a 10% validation split with early stopping. We compare against MAS-GPT (Ye et al., 2025), which generates multi-agent systems via code generation using a fine-tuned Qwen2.5-Coder-32B model.

Our LoRA SFT on the Qwen2.5-Coder-14B-Instruct model achieves 57.57% accuracy on GAIA validation, surpassing both MAS-GPT (23.03%) and a GRPO-augmented Llama-3-8B variant (31.52%). The result indicates that supervised learning on high-quality generation traces is more effective than reinforcement learning for this task, and that the meta-agent's workflow-design capability transfers cleanly to compact models.

Table 2. Distillation of workflow generation to smaller models. Accuracy (%) on GAIA validation across difficulty levels. GPT-5-mini workers.

METHOD	L1	L2	L3	AVG
MAS-GPT	32.10	20.93	11.54	23.03
LLAMA-3-8B (GRPO)	47.17	26.74	15.38	31.52
GRAPHGEN-14B (OURS)	**64.15**	**58.14**	**42.31**	**57.57**

4.5. Ablation: Two-Stage vs Unified Generation

We ablate the two-stage decomposition by comparing against unified generation, which produces agents and topology in a single pass (Table 3). With Gemini-2.5-flash workers on GAIA validation, two-stage outperforms unified by 3 percentage points (32.12% vs 29.09%), confirming that separating capability definition from coordination design improves workflow quality. The gap is most pronounced on Level 3 tasks (19.23% vs 11.54%), where complex multi-modal coordination benefits most from explicit reasoning about inter-agent dependencies.

Table 3. Ablation: Two-Stage vs Unified generation on GAIA validation. Accuracy (%). Gemini-2.5-flash workers.

STRATEGY	L1	L2	L3	AVG
TWO-STAGE	**41.51**	**30.23**	**19.23**	**32.12**
UNIFIED	35.85	**30.23**	11.54	29.09

4.6. Generation Strategy Comparison

We further compare TSAS's two-stage default against alternative generation strategies that encode different architectural biases: hierarchical (a coordinator with worker agents), decentralized (peer agents without hierarchy), ReAct-oriented (specialized archetypes inspired by ReAct (Yao et al., 2023)), and a single-agent baseline. Table 4 reports results with GPT-5-mini workers on GAIA validation.

Table 4. Accuracy across generation strategies on GAIA validation (165 tasks). GPT-5-mini workers.

STRATEGY	L1	L2	L3	AVG
TWO-STAGE (TSAS)	**69.81**	**46.51**	23.08	**50.30**
DECENTRALIZED	52.83	45.35	15.38	43.03
REACT-ORIENTED	52.83	36.47	23.08	39.63
SINGLE AGENT	52.83	38.37	**26.92**	41.21

The two-stage default is the clear winner on average (50.30%), outperforming decentralized (43.03%) by over 7

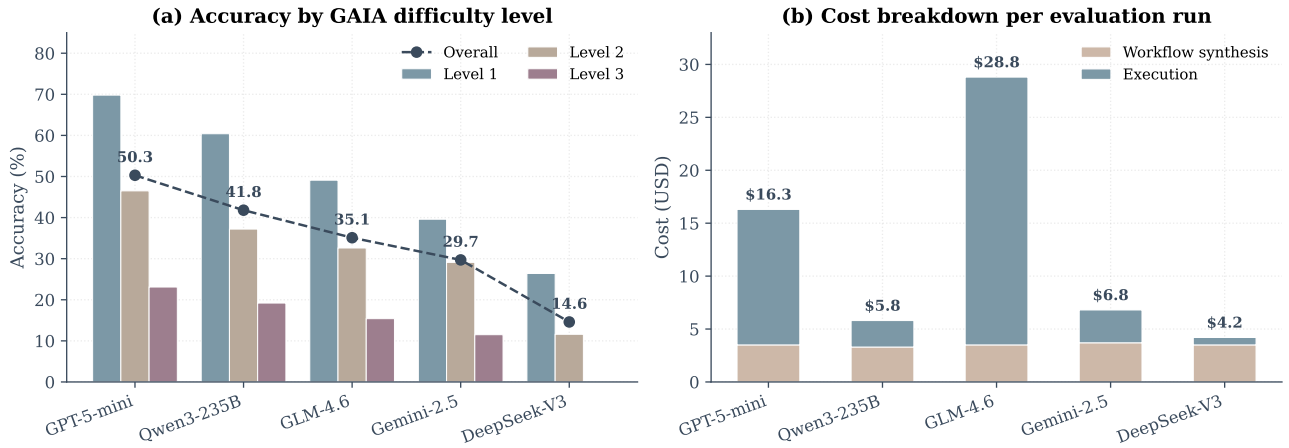

Figure 1. TSAS performance across five executor backbones on GAIA validation (165 tasks). **(a)** Accuracy by difficulty level, with the dashed line indicating overall accuracy. **(b)** Cost per evaluation run, decomposed into workflow synthesis (meta-agent calls) and execution (agent tool use and reasoning).

percentage points and the ReAct-oriented variant by more than 10. The single-agent baseline achieves competitive overall accuracy (41.21%) and the highest Level 3 accuracy (26.92%), suggesting that for the most complex multimodal tasks, concentrated reasoning within a single context can avoid coordination overhead. This provides direct evidence for the *adaptive* part of our design: the optimal coordination topology depends on task complexity, and the meta-agent's ability to allocate the right number of agents per task is itself a primary lever for performance.

4.7. Cost of Post-Generation Refinement

A common assumption in automated MAS design is that post-generation refinement—iterative regeneration guided by LLM-as-Judge feedback, evolutionary search, or meta-heuristic optimization—converts additional compute into accuracy gains. We empirically test this assumption on GAIA validation, comparing the single-pass TSAS baseline against three families of refinement.

Iterative regeneration. The meta-agent regenerates workflows over multiple rounds, with an LLM-as-Judge scoring each attempt's execution trace and providing architectural feedback. We evaluate two judge configurations against the no-refinement baseline (Table 5). Iterative regeneration with Claude Sonnet 4 as judge yields +6.67 pp in accuracy (50.30% $\rightarrow$ 56.97%) but at \$940.62 per run—58$\times$ the baseline cost. Substituting Gemini-2.5-Flash as the judge actively degrades performance, dropping accuracy by 11.51 pp (50.30% $\rightarrow$ 38.79%) at 7.7$\times$ cost: less accurate architectural feedback drives the meta-agent to suboptimal redesigns over successive iterations.

Evolutionary search. We additionally evaluated a population-based evolutionary approach with dedicated LLM-driven mutation and crossover operators applied to a meta-agent–generated seed, and selection guided by an LLM-as-Judge ensemble acting as surrogate fitness (Yuan et al., 2024; Liu et al., 2024). Because population-based optimization is prohibitively expensive on the full 165-task validation set, we report results on smaller subsets at matched configurations (population N=3, generations G=3, $p_m = p_c = 1.0$, GPT-5-mini workers). On a 30-task subset with a Gemini-2.5-Flash judge ensemble, accuracy was unchanged from the unoptimized baseline (46.67% in both cases) at 6–9$\times$ the baseline cost and 3–4$\times$ the wall-clock time. On a 14-task subset with a 2-judge ensemble we further compared judge models (Table 6): with Claude Sonnet 4, accuracy improved from 57.1% to 71.4% (+14.3 pp) at a $\sim$71$\times$ cost multiplier over the same-subset baseline, exceeding even the 58$\times$ overhead of iterative regeneration; with Gemini-2.5-Flash, accuracy dropped to 50.0% ($-$7.1 pp) at $\sim$8$\times$ cost.

Table 5. Cost of iterative regeneration on GAIA validation. The baseline is single-pass TSAS with GPT-5-mini workers.

CONFIGURATION	ACCURACY (%)	COST ($)
SINGLE-PASS (BASELINE)	50.30	16.31
+ SONNET 4 JUDGE	**56.97**	940.62
+ GEMINI-2.5-FLASH JUDGE	38.79	124.94

Metaheuristic search. We additionally evaluated Variable Neighborhood Search (VNS) (Hansen et al., 2017) and Variable Landscape Search (VLS) (Mussabayev & Mussabayev, 2025) over the agent pool, with neighborhoods defined by interpretable operators (agent permutation, instruction refinement, tool-set modification). The best variant (VLS with supervised judge) reaches 49.09% on GAIA validation—

Table 6. Cost of population-based evolutionary search on a 14-task GAIA validation subset. Working agents: GPT-5-mini; population N=3, generations G=3, $p_m = p_c = 1.0$; 2-judge ensemble.

CONFIGURATION	ACCURACY (%)	COST ($)
SINGLE-PASS (BASELINE)	57.1	1.12
+ GEMINI-2.5-FLASH JUDGE	50.0	9.02
+ SONNET 4 JUDGE	**71.4**	79.81

below the single-pass baseline (50.30%)—without offering meaningful cost savings.

Implication. Across all three families, single-pass synthesis Pareto-dominates on the cost-accuracy frontier: refinement either erodes accuracy (Gemini judges, VNS) or yields gains at 58–71× cost (Sonnet judges). The meta-agent's first-pass architectural inductive bias appears to be the primary driver of accuracy; subsequent refinement perturbs configurations more than it improves them. This supports TSAS's on-demand, single-pass design over inference-time refinement as a scaling axis.

4.8. Prompt Refinement

We additionally evaluate the optional HyPE prompt refinement (Section 3.6). On GAIA validation with GPT-5-mini workers, HyPE yields a +4.25 pp accuracy gain (50.30% → 54.55%) and simultaneously reduces inference cost by 15.1% ($16.31 → $13.84), as refined prompts produce more focused agent behavior with fewer unnecessary tool calls.

5. Discussion and Limitations

Scope: modalities and embodiment. GAIA spans diverse modalities accessed via tools—web search, image and audio understanding, structured file processing, code execution—and TSAS coordinates which agent handles which modality through which tool. The framework operates at the level of workflow orchestration rather than perception itself; it does not directly produce vision or audio embeddings, and continuous perception-action loops typical of embodied control are outside our current scope. Extending TSAS to embodied settings would require integrating perception-grounded executor agents while preserving the meta-agent's adaptive workflow synthesis.

Static workflows. Once generated, the workflow structure remains fixed during execution. An agent that discovers the task requires capabilities not present in the current pool cannot recruit additional agents or modify the topology. Incorporating runtime adaptation would address this limitation, though it introduces challenges around coordination overhead and stability.

Acyclic structure. The current framework generates DAGs, which preclude iterative refinement loops where agents revisit earlier stages based on downstream feedback. Extending the framework to support bounded recursion while guaranteeing termination is a natural next step.

6. Conclusion

We presented TSAS, a framework that generates task-specific multi-agent workflows on demand through meta-agent reasoning. The two-stage decomposition—first synthesizing a pool of specialized agents, then constructing a directed graph that defines their coordination—achieves 47.18% on the GAIA benchmark while reducing cost by 61% compared to autonomous systems with similar performance. TSAS-generated workflows consistently outperform autonomous agents built on more capable backbone models, suggesting that explicit decomposition provides architectural inductive bias that implicit reasoning struggles to discover. Workflow generation transfers effectively to smaller models: GraphGen-14B, fine-tuned on demonstrations from the meta-agent, achieves 57.57% on GAIA validation, more than doubling prior fine-tuned baselines without requiring reinforcement learning. Together, these results indicate that the meta-agent's coordination reasoning can be both deployed in a training-free regime on flagship models and distilled into compact models for efficient inference, opening a practical path for deploying multimodal agentic systems where full autonomy is undesirable or cost-prohibitive. A framework building on the methods presented here is publicly available.[2]

Acknowledgments

This work supported by the Ministry of Economic Development of the Russian Federation (IGK 000000C313925P4C0002), agreement No139-15-2025-010

Impact Statement

This paper develops methods for generating multi-agent systems that make coordination structure explicit and inspectable before execution, which we view as broadly beneficial for deployments where transparency matters. As with any automation framework, deployment in high-stakes settings should include human oversight of generated workflows; we make no claims that the meta-agent's outputs are correct by construction.

[2] https://github.com/ITMO-NSS-team/FEDOT.MAS

LLM Usage Disclosure

Writing. LLM assistants were used for proofreading, grammar correction, and rephrasing at the sentence level. The structure, technical claims, experimental design, and final wording of all sections are authored by the human authors.

Idea generation. LLM assistants were not used to generate research hypotheses, design baselines, select benchmarks, or interpret experimental results. All scientific content reflects the authors' own reasoning and analysis.

References

AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

Daniel Han, M. H. and team, U. Unsloth, 2023. URL http://github.com/unslothai/unsloth.

Hansen, P., Mladenović, N., Todosijević, R., and Hanafi, S. Variable neighborhood search: basics and variants. *EURO Journal on Computational Optimization*, 5(3):423–454, 2017. ISSN 2192-4406. doi: https://doi.org/10.1007/s13675-016-0075-x. URL https://www.sciencedirect.com/science/article/pii/S2192440621000873.

Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Wang, J., Zhang, C., Wang, Z., Yau, S. K. S., Lin, Z., Zhou, L., Ran, C., Xiao, L., Wu, C., and Schmidhuber, J. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Hu, S., Lu, C., and Clune, J. Automated design of agentic systems, 2025. URL https://arxiv.org/abs/2408.08435.

Hui, B., Yang, J., Cui, Z., Yang, J., Liu, D., Zhang, L., Liu, T., Zhang, J., Yu, B., Dang, K., et al. Qwen2. 5-coder technical report. *arXiv preprint arXiv:2409.12186*, 2024.

Kulin, N., Zhuravlev, V., Khairullin, A., Sitkina, A., and Muravyov, S. Coolprompt: Automatic prompt optimization framework for large language models. In *2025 38th Conference of Open Innovations Association (FRUCT)*, pp. 158–166. IEEE, 2025.

Li, G., Hammoud, H. A. A. K., Itani, H., Khizbullin, D., and Ghanem, B. Camel: Communicative agents for "mind" exploration of large language model society. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Li, S., Liu, Y., Wen, Q., Zhang, C., and Pan, S. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. *arXiv preprint arXiv:2507.18224*, 2025a.

Li, Y. et al. Agentswift: Efficient llm agent design via value-guided hierarchical search, 2025b.

Li, Z. et al. Autoflow: Automated workflow generation for large language model agents, 2024.

Liu, S., Chen, C., Qu, X., Tang, K., and Ong, Y.-S. Large language models as evolutionary optimizers, 2024. URL https://arxiv.org/abs/2310.19046.

Mialon, G., Fourrier, C., Swift, C., Wolf, T., LeCun, Y., and Scialom, T. Gaia: a benchmark for general ai assistants, 2023. URL https://arxiv.org/abs/2311.12983.

Mussabayev, R. and Mussabayev, R. Variable landscape search: A novel metaheuristic paradigm for unlocking hidden dimensions in global optimization. *Mathematics*, 13(22), 2025. ISSN 2227-7390. doi: 10.3390/math13223667. URL https://www.mdpi.com/2227-7390/13/22/3667.

Niu, B. et al. Flow: Modularized agentic workflow automation. In *The Thirteenth International Conference on Learning Representations*, 2025.

Shang, Y. et al. Agentsquare: Automatic llm agent search in modular design space, 2024.

Su, J. et al. Debflow: Automating agent creation via agent debate, 2025.

Wang, S. et al. Anymac: Cascading flexible multi-agent collaboration via next-agent prediction, 2025a.

Wang, Y. et al. Scoreflow: Mastering llm agent workflows via score-based preference optimization, 2025b.

Wang, Z., Wang, Y., Liu, X., Ding, L., Zhang, M., Liu, J., and Zhang, M. Agentdropout: Dynamic agent elimination for token-efficient and high-performance llm-based multi-agent collaboration. *arXiv preprint arXiv:2503.18891*, 2025c.

Wooldridge, M. *An Introduction to MultiAgent Systems*. John Wiley & Sons, 2009.

Wu, Q., Bansal, G., Zhang, J., Wu, Y., Li, B., Zhu, E., Jiang, L., Zhang, X., Zhang, S., Liu, J., Awadallah, A. H., White, R. W., Burger, D., and Wang, C. Autogen: Enabling next-gen llm applications via multi-agent conversation, 2023. URL https://arxiv.org/abs/2308.08155.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Ye, R. et al. Mas-gpt: Training llms to build llm-based multi-agent systems, 2025.

Yuan, S. et al. Evoagent: Towards automatic multi-agent generation via evolutionary algorithms, 2024.

Zhang, G., Yue, Y., Li, Z., Yun, S., Wan, G., Wang, K., Cheng, D., Yu, J. X., and Chen, T. Cut the crap: An economical communication pipeline for llm-based multi-agent systems. *arXiv preprint arXiv:2410.02506*, 2024a.

Zhang, G., Yue, Y., Sun, X., Wan, G., Yu, M., Fang, J., Wang, K., Chen, T., and Cheng, D. G-designer: Architecting multi-agent communication topologies via graph neural networks. *arXiv preprint arXiv:2410.11782*, 2024b.

Zhang, G. et al. Evoflow: Evolving diverse agentic workflows on the fly, 2025a.

Zhang, G. et al. Multi-agent architecture search via agentic supernet, 2025b.

Zhang, J. et al. Aflow: Automating agentic workflow generation, 2024c.

Zhuge, M. et al. Gptswarm: Language agents as optimizable graphs. In *Forty-first International Conference on Machine Learning*, 2024.