MATHTRIX

Research Multi-Agent Architecture

5 Agents + CODE_TOOL + Navigation Scheme

CORE PRINCIPLE

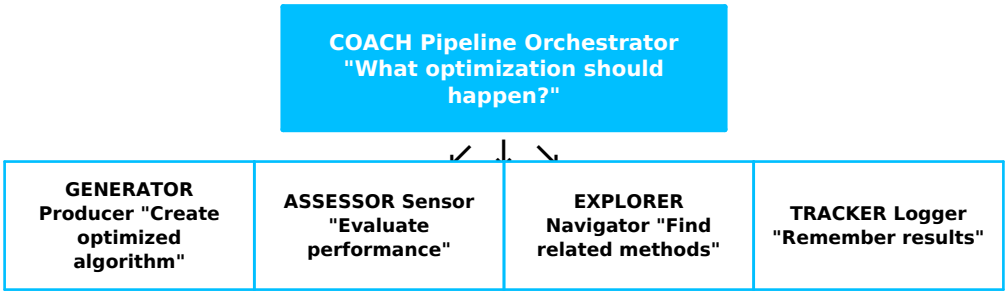
"The 5 agents are topological primitives. CODE_TOOL is a specialized tool (not 6th agent). Experiences are pipelines (compositions), not new agents."

Agent	Topology	Cognitive Role	Operation
COACH	Hub	Executive	"What should happen?" (Control)
GENERATOR	Producer	Creation	"Create something" (Production)
ASSESSOR	Sensor	Evaluation	"How is it performing?" (Measurement)
EXPLORER	Navigator	Search	"Find related methods" (Retrieval)
TRACKER	Logger	Memory	"Remember this" (Persistence)

These 5 agents are UNIVERSAL — same for Education AND Research. Any experience = composition of these primitives.

1. Agent Topology — Visual Hierarchy

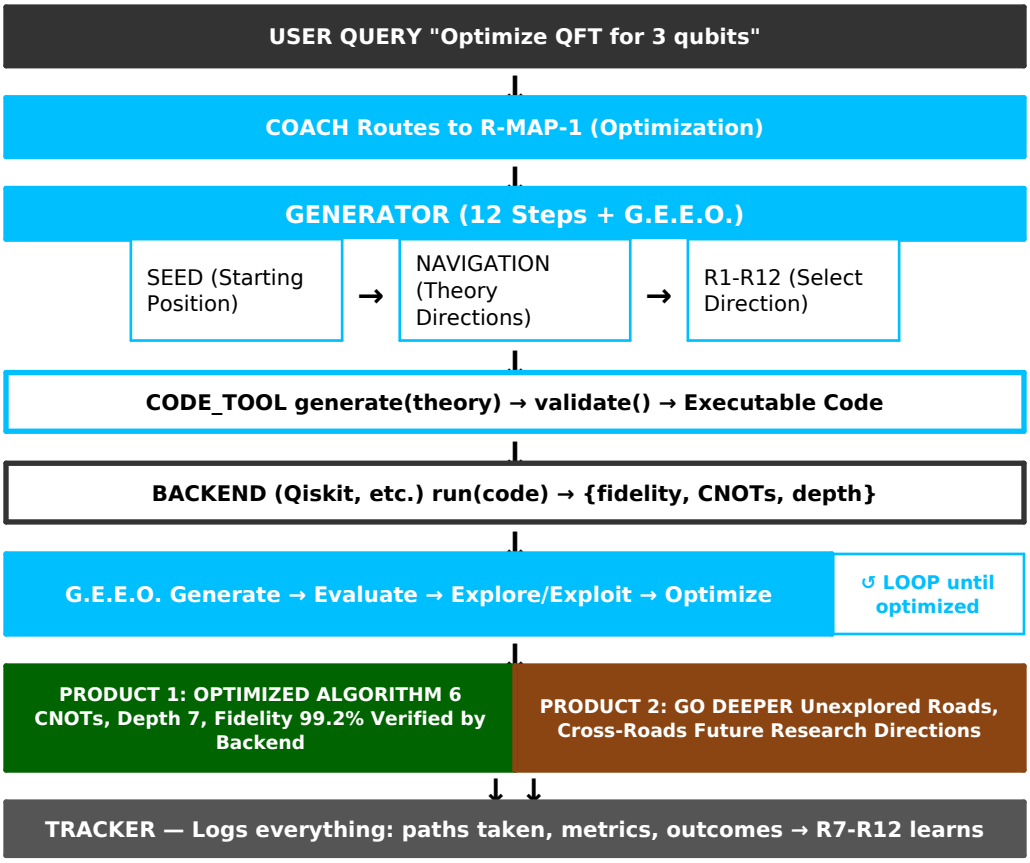
1.1 Research Agent Topology Diagram



— — → Pipeline chaining (COACH routes to workers)
· · · · → All workers feed TRACKER (results logged)

1.2 Research Axis Complete Scheme

The Research Axis flow from query to verified optimized algorithm:



1.3 Research Pipelines (Experiences = Agent Compositions)

Experience	Pipeline	Loop?
Optimization (R-MAP-1)	GENERATOR → TRACKER	No
Iterative Optimization	GENERATOR → ASSESSOR → GENERATOR	Yes
Simulation Protocol (R-MAP-2)	GENERATOR (replay) → TRACKER (logs)	No
Navigation (R-MAP-3)	EXPLORER	User-driven
Exploration-First	EXPLORER → GENERATOR → TRACKER	No
Benchmark Challenge	ASSESSOR → GENERATOR → ASSESSOR	Yes
Historical Analysis	TRACKER → ASSESSOR → GENERATOR	No
Ablation Study	[GENERATOR → ASSESSOR] × N	No

Adding New Experience: 0 New Agents

```
# coach.py — just add one line PIPELINES = { "optimize": [Generator, Tracker], "iterate": [Generator, Assessor, Generator], "explore": [Explorer, Generator, Tracker], "pareto": [Generator, Assessor, Explorer], # ← NEW LINE } No new agent. Just a new pipeline using existing primitives.
```

2. The 5 Agents: Education vs Research Context

Agent	Education Context	Research Context
COACH (Hub)	Routes to LEARNING pipelines • Classifies learner intent • Selects teaching approach	Routes to OPTIMIZATION pipelines • Classifies research intent • Selects optimization approach
GENERATOR (Producer)	Creates CURRICULUM content • 12-step pipeline • Uses Education Seeds • Calls LLM for content	Creates OPTIMIZED algorithms • 12-step pipeline • Uses Research Seeds • Calls Backend for execution
ASSESSOR (Sensor)	Evaluates LEARNER state • Knowledge gaps • Difficulty level	Evaluates ALGORITHM performance • Metric gaps (fidelity, gates) • Optimization potential
EXPLORER (Navigator)	Navigates KNOWLEDGE graph • Related topics • Reverse simulation (WHY)	Navigates ALGORITHM graph • Related algorithms • Method tracing (WHY worked)
TRACKER (Logger)	Tracks LEARNING progress • XP per cognitive spine • Achievements, badges	Tracks OPTIMIZATION history • Results per metric • Best solutions, method effectiveness

3. Research MAPs (Experiences)

3.1 The 4 Core Research MAPs

R-MAP	Experience	Primary	Supporting
R-MAP-1 Optimization Generation	Query → Optimized Algorithm (Black Box)	GENERATOR	ASSESSOR, COACH
R-MAP-2 Simulation Protocol	See HOW Generator arrived at answer (Transparent Journey)	GENERATOR	TRACKER, EXPLORER
R-MAP-3 Navigation (Graph)	Algorithm → Related Algorithms "What else can I optimize?"	EXPLORER	COACH
R-MAP-4 Metrics & History	Results, Comparisons, Leaderboards	TRACKER	COACH

3.2 Additional Research Experiences (Same 5 Agents)

Experience	Pipeline	Loop?	Description
Iterative Optimization	GENERATOR → ASSESSOR → GENERATOR	Yes	Multi-round refinement
Exploration-First	EXPLORER → GENERATOR → TRACKER	No	Find methods, then apply
Benchmark Challenge	ASSESSOR → GENERATOR → ASSESSOR	Yes	Beat a baseline score
Historical Analysis	TRACKER → ASSESSOR → GENERATOR	No	Review past, suggest next
Ablation Study	[GENERATOR → ASSESSOR] × N	No	Test variants, compare
Pareto Exploration	GENERATOR → ASSESSOR → EXPLORER	Yes	Find trade-off frontier

4. WHERE the Navigation Scheme Exists

The Navigation Scheme is STORED in the Math Library as a queryable GRAPH:

Level	Count	Storage	Data Structure
PILLARS	10	math_library.pillars	pillar_id, name, description, parent_pillars
ROADS	152	math_library.roads	road_id, pillar_id, sequence, prerequisites
CROSS-ROADS	~50	math_library.cross_roads	xroad_id, from_pillar, to_pillar, bridge_concepts
PATH STEPS	~750	math_library.path_steps	step_id, road_id, position, theorem/method
CONCEPTS	~2000	math_library.concepts	concept_id, name, related_steps
TASKS	~500	math_library.tasks	task_id, operation_type, applicable_concepts

4.1 Graph Relationships

Relationship	Type	Enables
Pillar → Roads	1:N	Find all Roads in a domain
Road → Path Steps	1:N (ordered)	Get methods within a theory
Pillar ↔ Pillar (Cross-Roads)	N:N	Cross-domain exploration
Path Step → Concepts	N:N	Understand what method manipulates
Concept → Tasks	N:N	Find applicable operations

5. HOW Navigation Scheme Integrates with Generator

The key insight: G.E.E.O. (Step 5) QUERIES the Navigation Scheme to guide its exploration.

5.1 Enhanced Generator (Research) — Seeds + Navigation

Step	Action	Navigation Source
1. CLASSIFY	Parse optimization query	—
2. ATTACH	Load Research Seed	Seed Block 5 (starting position)
3. ENFORCE	Validate constraints	—
4. PROJECT	Map to optimization space	—
5. G.E.E.O. (Enhanced)	EXPLORE phase queries Navigation: • "What adjacent Roads exist?" • "What Cross-Roads connect here?" • "What Path Steps haven't we tried?"	MATH LIBRARY QUERIES get_adjacent_roads() get_cross_roads() get_untried_steps() get_tasks()
6-12	HEART, Backend, Quality, etc.	—

5.2 G.E.E.O. Phases with Navigation

Phase	Without Navigation	With Navigation Scheme
GENERATE	Use Seed's methods only	+ Query: "What Path Steps apply?"
EVALUATE	Score candidates (metrics)	Same — metric-based
EXPLORE	Random (ε-greedy) No direction	DIRECTED: Query adjacent Roads, Cross-Roads, untried Path Steps
EXPLOIT	Refine best locally	+ Query: "What Tasks apply?"
OPTIMIZE	Select best	Same — metric-based

6. The Combined Architecture

6.1 Component Roles

Component	Location	Role
Research Seed (Block 5: Math Tree)	Loaded at Step 2	STARTING POSITION: Initial Pillar, Road, Path Steps → WHERE TO START
Navigation Scheme (Math Library Graph)	Queried at Step 5	EXPLORATION MAP: Adjacent Roads, Cross-Roads, untried methods → WHERE TO GO NEXT
R1-R6 Bandits (RL System)	Used at Step 5	SELECTION: Which navigation option to try? → HOW TO CHOOSE
TRACKER History	Updated at Step 12	LEARNING: Which Roads led to success? → WHAT WE LEARNED

6.2 Navigation Queries

Query	Input	Output	Used In
get_adjacent_roads(road_id)	Current Road	Nearby Roads	EXPLORE
get_cross_roads(pillar_id)	Current Pillar	Cross-domain connections	EXPLORE
get_path_steps(road_id)	Road	Methods in this theory	GENERATE
get_untried_steps(road, tried[])	Road + history	Unexplored methods	EXPLORE
get_tasks(concept_id)	Concept	Applicable operations	EXPLOIT

6.3 CODE_TOOL: The Theory → Code Bridge

Critical question: How does a mathematical theory direction become executable code?

Component	Role	Why Not 6th Agent?
CODE_TOOL (Specialized Tool)	TRANSLATOR: Theory → Executable Code • generate(theory, target, constraints) → code • validate(code) → {valid, errors} • run(code, backend) → {metrics}	• 5 agents remain topological primitives • Tool is stateless (uses GENERATOR state) • Swappable without changing architecture • GENERATOR decides WHAT, tool does HOW

6.4 Research Axis Complete Flow

Navigation → R1-R12 → CODE_TOOL → Backend → G.E.E.O. → Loop

The complete Research Axis flow integrates all components:

1. Navigation Scheme provides theory directions (Roads, Cross-Roads, Path Steps) 2. R1-R12 selects which direction to try (UCB1 bandit) 3. CODE_TOOL.generate() translates theory into executable code 4. CODE_TOOL.validate() checks syntax and constraints 5. Backend.run() executes code and returns metrics 6. G.E.E.O. evaluates results and decides next action 7. Loop until optimized or budget exhausted

7. Concrete Example: QFT with Full Navigation + CODE_TOOL

7.1 Navigation Resources Used:

6 Roads	5 Cross-Roads	7 Concepts
CR-ALG-2 (Linear Computation) CR-ALG-4 (Representation) CR-AN-10 (Harmonic Analysis) CR-AN-14 (Complex Structures) A1 (Numerical LinAlg) OPT2 (Combinatorial Opt)	XR-ALG-AN (Algebra↔Analysis) XR-ALG-APPL (Algebra↔Applied) XR-AN-APPL (Analysis↔Applied) XR-APPL-OPT (Applied↔Optimization) XR-ALG-OPT (Algebra↔Optimization)	DFT_2 matrix → Hadamard gate Root of unity ω^k → CP gates Tensor factorization → Recursive Butterfly op → H + CP structure Bit reversal → SWAP gates Parallel independence → Layers Cyclic group character → Patterns

7.2 Complete Flow with CODE_TOOL:

Phase	Action	CODE_TOOL + Backend
ATTACH	Load QFT Research Seed Seed Block 5: • Pillar: Analysis, Algebra • Road: CR-AN-10 • Cross-Roads: XR-ALG-AN	Baseline (standard QFT): 12 CNOTs, Depth 15 Fidelity 100%
G.E.E.O. #1 GENERATE	Navigation: get_path_steps("CR-AN-10") → [DFT, FFT, Twiddle, Butterfly] R1-R12: "Apply FFT restructuring"	CODE_TOOL.generate() theory="FFT butterfly" → Qiskit code Backend.run() 10 CNOTs (17% ↓), Depth 12 (20% ↓)
G.E.E.O. #1 EXPLORE	Navigation: get_cross_roads("CR-AN-10") → XR-ALG-AN, XR-AN-APPL R3 (UCB1): EXPLORE via XR-ALG-AN	(waiting for theory)
G.E.E.O. #2 GENERATE	Navigation: get_path_steps("CR-ALG-4") → [Maschke, Schur, Character] R1-R12: "Apply Z_8 cyclic symmetry"	CODE_TOOL.generate() theory="Z_8 cyclic" → optimized code Backend.run() 7 CNOTs (42% ↓), Depth 10 (33% ↓)
G.E.E.O. #2 EXPLOIT	R5 (Thompson): EXPLOIT Navigation: get_tasks("CR-ALG-4") → [merge_phases, pattern_reuse] R1-R12: "Apply phase merging"	CODE_TOOL.generate() theory="phase merging" → final code Backend.run() 6 CNOTs (50% ↓) Depth 7 (53% ↓) Fidelity 99.2%
OUTCOME	TRACKER logs path: CR-AN-10 → XR-ALG-AN → CR-ALG-4 → XR-ALG-OPT → OPT2 R7-R12 learns: "For QFT: cross to Algebra early"	Product 1: Algorithm 6 CNOTs, Depth 7, F=99.2% Verified by Backend Product 2: GO DEEPER Unexplored: Lie algebra, approximate QFT, KAK

8. Complete Research Architecture

8.1 Agent-to-MAP Coverage

MAP	Experience	Primary	Uses Navigation?
R-MAP-1	Optimization Generation	GENERATOR	YES — G.E.E.O. queries graph
R-MAP-2	Simulation Protocol	GENERATOR	YES — replays journey through graph
R-MAP-3	Algorithm Navigation	EXPLORER	YES — displays graph to user
R-MAP-4	Metrics & History	TRACKER	No — metrics only

8.2 Shared State Architecture

MATH LIBRARY (Navigation Scheme)	DOMAIN LIBRARY
• 6-Level Hierarchy Graph • 10 Pillars, 152 Roads • ~50 Cross-Roads • ~750 Path Steps	• Templates (Q1-Q5) • Constraints & Bounds • HEART 61 Templates • Backend Patterns
RESEARCH SEEDS	**RESEARCH STATE**
• ~1000 Seeds per domain • Block 5: Starting Position • Pre-encoded Roads/Path Steps	• Current problem context • Best results so far • Tried methods history • Session context

9. Summary

Question	Answer
How many agents?	5 — COACH, GENERATOR, ASSESSOR, EXPLORER, TRACKER (same as Education)
How many experiences?	Unlimited — 4 core MAPs + any pipeline composition
WHERE is Navigation Scheme?	Math Library — tables for Pillars, Roads, Cross-Roads, Path Steps, Concepts, Tasks
HOW does it integrate?	G.E.E.O. queries it — during EXPLORE and EXPLOIT phases
What does Seed provide?	Starting Position — initial Pillar, Road, Path Steps
What does Navigation provide?	Exploration Map — adjacent Roads, Cross-Roads, untried methods
What do R1-R6 provide?	Selection — which navigation option to try
What does CODE_TOOL do?	Theory → Code — translates mathematical direction into executable code
What does Backend do?	Verification — executes code and returns metrics
What are the 2 Products?	Product 1: Optimized Algorithm (verified by Backend) Product 2: GO DEEPER (unexplored paths for future research)

THE COMPLETE FORMULA

5 AGENTS + CODE_TOOL + BACKEND = AUTOMATED OPTIMIZATION

Navigation → R1-R12 → CODE_TOOL → Backend → G.E.E.O. loops until optimized

Navigation Scheme provides DIRECTED search. CODE_TOOL translates theory into executable code. Backend verifies it works. G.E.E.O. evaluates and loops.