Deep Hierarchical Reinforcement Learning: Architectures, Algorithms, and Emergent Paradigms (2024-2025)

1. Executive Summary and Theoretic Framework

The pursuit of Artificial General Intelligence (AGI) is fundamentally a problem of scaling decision-making across disparate temporal and spatial resolutions. While standard Reinforcement Learning (RL) has demonstrated remarkable proficiency in reactive environments—where the mapping between observation and optimal action is direct and immediate—it historically falters in domains requiring long-horizon planning, strategic reasoning, and the coordination of abstract goals with primitive actuation. The years 2024 and 2025 have witnessed a structural transformation in the field of Deep Hierarchical Reinforcement Learning (HRL), marking a departure from the strict feudal architectures of the past decade toward flexible, neuro-symbolic, and language-grounded hierarchies.

This report provides an exhaustive analysis of the state-of-the-art in Deep HRL as of early 2025. It synthesizes research across theoretical advancements in non-stationarity mitigation, architectural innovations in graph-based and transformer-based controllers, and the paradigm-shifting integration of Large Language Models (LLMs) as high-level planners.

The central thesis emerging from this analysis is that HRL is transitioning from a "decomposition problem"—how to break a task into subtasks—to an "alignment problem"—how to align the semantic priors of foundation models with the physical constraints of embodied agents. We observe three dominant trends:

1. **Context-Decoupled Reasoning:** The emergence of architectures like CoDA and GLIDER that solve the "context explosion" problem by physically separating strategic planning from execution details.[1]
2. **Preference-Driven Stability:** The replacement of brittle, manually shaped rewards with robust preference optimization frameworks (PIPER, DIPPER) to address the inherent non-stationarity of learning in a hierarchy.[3]
3. **Emergent Hierarchy:** The discovery that hierarchical reasoning is not merely an imposed architectural constraint but an emergent property of sufficiently advanced credit assignment mechanisms (HICRA) in large-scale models.[5]

---

2. The Theoretical Imperative: Why Hierarchy?

To understand the breakthroughs of 2025, one must first rigorously define the limitations they address. Standard RL formulations, modeled as Markov Decision Processes (MDPs), struggle with the "Curse of Dimensionality" in time. As the time horizon T of a task grows, the search space for a sequence of actions grows exponentially ($|A|^T$). Furthermore, the "Vanishing Gradient" of reward signals implies that an action taken at $t=0$ becomes increasingly difficult to credit for a reward received at $t=10,000$.[6]

Hierarchical Reinforcement Learning addresses these scaling laws by introducing **Temporal Abstraction**. By decomposing a task into subroutines (or "options"), the effective horizon of the high-level policy is reduced from T to T/k, where k is the average duration of a subroutine. This transforms the problem from a single MDP into a Semi-Markov Decision Process (SMDP) or a hierarchy of MDPs.

However, classic HRL introduces its own pathologies:

- **Non-Stationarity:** The lower-level policy (Worker) is learning to execute subgoals at the same time the high-level policy (Manager) is learning to propose them. From the Manager's perspective, the environment's transition dynamics are constantly shifting, destabilizing learning.[8]
- **Subgoal Representation:** Defining the "language" in which the Manager commands the Worker is non-trivial. Discrete spaces are limited; continuous spaces are hard to explore.
- **Reward Design:** How does one reward the Worker for completing a subgoal if the subgoal itself contributes to a larger failure?

The following sections detail how the research of 2024-2025 has systematically dismantled these barriers.

---

3. The Neuro-Symbolic Turn: Large Language Models as Hierarchical Controllers

The most significant disruption in the HRL landscape is the integration of Large Language Models (LLMs). LLMs provide a pre-trained "World Model" containing vast semantic knowledge about task decomposition, which was previously the hardest part for an HRL agent to learn from scratch. However, LLMs are not inherently embodied agents; they lack grounding in physical dynamics and suffer from context window limitations. The architectures of 2025 are defined by how they solve these specific deficits.

3.1 CoDA: The Context-Decoupled Hierarchical Agent

In long-horizon tasks, such as complex information retrieval or multi-step coding, an agent accumulates a massive history of observations, tool outputs, and intermediate thoughts. For a monolithic agent, this history quickly saturates the context window, leading to "Context Explosion." More critically, it leads to "Context Entanglement," where high-level strategic reasoning is polluted by the noise of low-level execution details.[1]

The CoDA Framework (Context-Decoupled Agent) [1] represents a fundamental re-architecture of the agentic memory model. It posits that efficient hierarchical reasoning requires *ignoring* information as much as retaining it. CoDA decomposes the decision-making process into two distinct roles served by a single LLM backbone, differentiated by their access to information.

3.1.1 Architectural Mechanism: The Strategic vs. Execution Context

CoDA enforces a strict informational barrier between the **Planner** and the **Executor**:

- **The Planner (Strategic Context):** This role operates at the highest level of abstraction. Its context window, the "Strategic Context," is deliberately sparse. It contains only the original user query, the high-level plan, and distilled summaries of completed sub-tasks. It *never* sees raw documents, error logs, or verbose tool outputs. Its function is solely to decompose the problem into the next logical sub-task, g_t.[1]
- **The Executor (Execution Context):** Upon receiving a sub-task g_t, the system instantiates an Executor. This role operates within a "Temporary Execution Context." It interacts with the environment (e.g., browsing the web, running code), processes the raw, noisy returns, and performs the necessary reasoning to satisfy the sub-task. Crucially, this context is **ephemeral**.
- **The Information Bottleneck:** Once the Executor completes its sub-task, it must synthesize its findings into a concise answer, a_t. Only this distilled answer a_t is passed back to the Planner. The Temporary Execution Context is then flushed/deleted.

This architecture mimics the human cognitive distinction between executive function (strategy) and motor control (execution). By abstracting away the execution details, the Planner effectively "sees" a simplified, stationary environment, allowing it to maintain coherence over horizons that would cause a standard LLM to hallucinate or lose focus.[12]

3.1.2 Planner-Executor Co-Optimization (PECO)

A critical innovation in CoDA is its training methodology. Simply prompting an LLM to play these roles is insufficient. CoDA introduces **PECO**, an end-to-end reinforcement learning framework.

- **Trajectory Generation:** During training, the system generates hierarchical trajectories where the Planner delegates to the Executor.
- **Composite Reward Function:** The system uses a multi-faceted reward that evaluates both the final answer quality (correctness) and the efficiency of the decomposition (number of steps).
- **Group-Level Credit Assignment:** Using **Group Relative Policy Optimization (GRPO)**, CoDA generates multiple diverse trajectories for the same query. It then updates the policy weights to favor those hierarchical decompositions that yielded the correct answer

most efficiently. This effectively "aligns" the Planner and Executor, teaching the Planner to ask for feasible sub-tasks and the Executor to provide useful summaries.[1]

Empirical Validation:
On complex multi-hop Question Answering (QA) benchmarks, CoDA demonstrated a performance improvement of up to 6.0% in accuracy over state-of-the-art monolithic baselines. More importantly, ablation studies revealed superior "long-context robustness." While monolithic agents like AutoRefine saw performance degrade by over 50% as the number of retrieved documents increased, CoDA's performance remained stable, validating the hypothesis that hierarchical context decoupling is essential for scaling inference.1

3.2 GLIDER: Grounding LLMs via Offline HRL

While CoDA addresses the context problem, it assumes the underlying LLM is capable of executing the sub-tasks if given the right context. In physical domains (Robotics, embodied agents), this assumption fails. LLMs often "hallucinate" actions that are semantically plausible (e.g., "pick up the hot soup") but physically disastrous (e.g., the robot hand has no heat shielding).

GLIDER (Grounding Language Models as Efficient Decision-Making Agents) [2] addresses this **Grounding Problem**. It serves as a bridge between the semantic intelligence of LLMs and the physical competence of classical Reinforcement Learning.

3.2.1 The Divide-and-Conquer Hierarchy

GLIDER implements a hierarchy that leverages the best of both worlds:

1. **High-Level Policy (LLM-based):** This layer exploits the LLM's common-sense reasoning to generate abstract plans. These are not merely text outputs but "chain-of-thought reasoning sub-tasks." The LLM decomposes a long-horizon goal (e.g., "Clean the kitchen") into a sequence of semantic subgoals (e.g., "Find the sponge," "Go to the sink," "Scrub the counter").
2. **Low-Level Controller (Offline RL):** This is the core innovation. Instead of asking the LLM to output motor torques or primitive actions, GLIDER delegates execution to a low-level policy trained via **Offline Reinforcement Learning**.

3.2.2 Task-Agnostic Skills and Parameter Efficiency

The low-level controller in GLIDER is trained on massive datasets of prior robot interactions to master a set of "task-agnostic" skills. These skills (e.g., navigation, grasping, manipulation) are general-purpose.

- **Transferability:** Because the low-level skills are not tied to a specific high-level objective, they possess strong transferability. If the agent is moved to a new environment, the LLM high-level policy can immediately re-plan using the same library of low-level skills.
- **Safety and Exploration:** Offline RL allows the agent to learn robust skills without the

risks associated with online exploration in physical environments. The hierarchy essentially confines the "exploration" to the semantic space (planning), while the physical interaction remains within the safety bounds of the pre-trained policy.[2]

Performance:
Experiments on benchmarks like ScienceWorld and ALFWorld demonstrate that GLIDER achieves consistent performance gains over flat LLM agents. The "divide-and-conquer" approach allows it to solve tasks requiring dozens of steps by breaking them into manageable units that the offline policy can reliably execute. This confirms that HRL is a necessary component for "grounding" foundation models in physical reality.2

3.3 HICRA: Emergent Hierarchies and Credit Assignment

A fundamental question in Deep Learning is whether hierarchies must be explicitly engineered (as in CoDA and GLIDER) or if they can *emerge* naturally from the optimization process. Research in 2025 has provided compelling evidence for the latter, specifically within the reasoning traces of RL-tuned LLMs.

HICRA (Hierarchy-Aware Credit Assignment) [5] explores the internal dynamics of reasoning. The authors posit that when an LLM is trained with Reinforcement Learning (e.g., via PPO or GRPO) to solve complex problems, it naturally develops a latent hierarchy of "strategic planning" vs. "procedural execution."

3.3.1 The Strategic Bottleneck and Planning Tokens

The core insight of HICRA is that not all tokens in a generated chain-of-thought are created equal.

- **Planning Tokens:** These are "fork points" or strategic pivots where the model decides on a high-level approach (e.g., "I will solve this by integration by parts"). These tokens are characterized by high **Semantic Entropy**—meaning there are multiple valid high-level strategies the model could choose, leading to high uncertainty in the strategic space.[5]
- **Procedural Tokens:** These are the deterministic steps that follow a plan (e.g., the algebraic manipulation after choosing integration). These tend to have lower entropy.

Standard RL methods apply credit assignment (reward/penalty) uniformly across the entire sequence. HICRA argues this is inefficient: if a math solution is wrong, it is likely due to a bad plan (the high-entropy fork), not a typo in the calculation.

3.3.2 The HICRA Algorithm

HICRA modifies the policy gradient update to concentrate optimization pressure on these high-impact planning tokens.

- **Mechanism:** The algorithm dynamically weights the gradient update based on the semantic role of the token. It identifies "strategic bottlenecks"—tokens that disproportionately influence the downstream trajectory—and amplifies the learning

signal for them.
- **Emergent Reasoning:** This targeted pressure accelerates the emergence of sophisticated reasoning patterns. The model learns "how to plan" faster because the RL feedback is directed at the planning steps rather than being diluted across the execution steps.[14]

Empirical Evidence:
Across multiple reasoning benchmarks, HICRA consistently outperforms strong baselines like standard GRPO. It achieves this by explicitly recognizing and reinforcing the hierarchical structure of thought, proving that Deep HRL principles apply as much to cognitive reasoning in transformers as they do to motor control in robots.13

---

4. Solving the Stability Crisis: Non-Stationarity and Subgoal Feasibility

While LLM-based hierarchies tackle semantic complexity, the application of HRL to continuous control (Robotics) faces a different, more mathematical beast: **Non-Stationarity**.

In a typical Manager-Worker hierarchy, the Manager learns to select subgoals ($g$) to maximize external reward ($R$), while the Worker learns to reach those subgoals ($g$) to maximize an intrinsic reward. The critical flaw is that the Worker is learning *simultaneously*. A subgoal that is "impossible" at Epoch 1 (yielding low reward for the Manager) might become "easy" at Epoch 100 (yielding high reward). This means the transition dynamics observed by the Manager ($P(s'|s,g)$) are constantly shifting, violating the stationary MDP assumption required for convergence.[8]

The year 2024 has seen a definitive shift away from trying to "fix" the reward function manually, moving instead toward **Preference-Based Learning**.

4.1 PIPER: Primitive-Informed Preference-based Learning

PIPER (Primitive-Informed Preference-based Hierarchical RL via Hindsight Relabeling) [3] offers a comprehensive solution to non-stationarity by redefining how the Manager receives feedback.

4.1.1 The Preference Paradigm

Instead of relying on a fixed reward function, PIPER uses preference learning. The agent compares two trajectories and decides which is "better."

- **Primitive-in-the-Loop (PiL):** Traditionally, preference learning (like RLHF) requires human feedback, which is slow and expensive. PIPER introduces an autonomous "Primitive-in-the-Loop" scheme. It uses the sparse signals from the environment (did the robot eventually succeed?) to generate synthetic preferences, allowing the agent to learn a high-level reward model without a human in the loop.[3]

- **Hindsight Relabeling:** To address sparse rewards, PIPER employs hindsight relabeling. If a trajectory failed to reach goal G but reached state S, the system relabels the goal as S and treats the trajectory as a success for that specific goal. This drastically improves the data efficiency of the preference model.[3]

4.1.2 Primitive-Informed Regularization

The most profound contribution of PIPER is Primitive-Informed Regularization. A common failure mode in HRL is the Manager generating "infeasible subgoals"—telling a robot to move its hand through a table.
PIPER conditions the high-level policy update on the Value Function of the Lower Level (V_{π_L}).

$$L_{reg} \approx - \log \pi_{High}(g|s) \cdot V_{\pi_L}(s, g)$$

Intuitively, $V_{\pi_L}(s, g)$ represents the Worker's current confidence in its ability to reach goal g from state s. By weighting the Manager's policy update with this value, PIPER forces the Manager to propose subgoals that are currently achievable by the Worker. As the Worker improves (and V_{π_L} increases for harder goals), the Manager is naturally permitted to select more ambitious subgoals. This creates an automatic, dynamic curriculum that explicitly synchronizes the learning rates of the two levels, solving the non-stationarity problem.18
Results:
In challenging sparse-reward environments like Franka Kitchen and Maze Navigation, where flat RL and standard HRL baselines often fail to make any progress (0% success), PIPER achieves success rates greater than 50%. This demonstrates that feasibility-aware regularization is a prerequisite for functional HRL in continuous domains.3

4.2 DIPPER: Direct Preference Optimization in Hierarchy

Following the success of PIPER, **DIPPER (Direct Preference Optimization for Primitive-Enabled HRL)** [4] streamlines the architecture by adapting **Direct Preference Optimization (DPO)**—a technique famous in LLM fine-tuning—to hierarchical control.

4.2.1 Bi-Level Optimization

DIPPER observes that learning a separate reward model (as in PIPER) adds complexity and potential error. Instead, DIPPER formulates the HRL problem as a bi-level optimization:

- **Upper Level:** Minimizes a DPO loss derived directly from trajectory preferences. This skips the reward modeling step entirely, optimizing the policy to satisfy preferences directly.
- **Lower Level:** Optimizes a standard RL objective to reach the subgoals set by the upper

level.[21]

4.2.2 Primitive-Enabled Reference Policy

DPO requires a "reference policy" (a baseline model) to prevent the agent from drifting too far and collapsing. DIPPER constructs a novel **Primitive-Enabled Reference Policy**. This reference policy is designed to inherently prefer feasible subgoals (based on the primitives' capabilities). By anchoring the DPO optimization to this feasibility-aware reference, DIPPER ensures that the Manager's exploration remains grounded in reality.[22]

Significance:
DIPPER represents the first successful application of DPO to Hierarchical Reinforcement Learning in robotics. It offers a mathematically elegant alternative to reward engineering, showing that the principles of alignment (from LLMs) are directly transferable to control theory.22

---

5. Structural Evolution: Graphs and Transformers

Beyond algorithms, the very *structure* of the hierarchical agent is evolving. The years 2024-2025 have moved beyond the simple "two-layer" (Manager-Worker) MLP networks toward architectures that reflect the complexity of the data they process.

5.1 Feudal Graph Reinforcement Learning (FGRL)

In complex physical systems (e.g., a humanoid robot), the "hierarchy" is not just temporal; it is structural. The left leg is independent of the right arm, but both must coordinate for walking. **Feudal Graph RL (FGRL)** [23] embeds this structural hierarchy directly into the agent's neural architecture.

5.1.1 Pyramidal Message Passing

FGRL represents the agent as a layered graph $\mathcal{G}^*$:

- **Leaf Nodes (Workers):** These correspond to the physical actuators/joints. They interact directly with the environment.
- **Intermediate Nodes (Sub-Managers):** These abstract nodes manage clusters of workers (e.g., a "Leg Manager" controlling the hip, knee, and ankle).
- **Root Node (Super-Manager):** The global strategist.

Communication occurs via a **Pyramidal Message Passing** protocol. The Super-Manager computes a high-level goal vector, which is propagated down the graph. Each Sub-Manager receives this message, integrates it with its local state, and generates sub-goals for its children. Conversely, state information propagates upward, but is compressed/abstracted at each level.[23]

5.1.2 Decoupled and Composable Learning

This graph structure allows for **Composable Policies**. Because each node is a modular policy, a "Leg Manager" trained in one task can theoretically be unplugged and reused in a different agent configuration. FGRL solves the "information bottleneck" of standard GNNs by ensuring that high-level planning signals (from the root) don't get drowned out by the noise of raw sensory data at the leaves. Empirical results on MuJoCo locomotion tasks show that FGRL outperforms standard message-passing baselines, particularly in tasks requiring global coordination of disparate limbs.[24]

5.2 Decision Transformer Hierarchical Reinforcement Learning (DT-HRL)

The **Decision Transformer (DT)** revolutionized RL by treating it as a sequence modeling problem (predicting the next action given history). **DT-HRL** [27] extends this paradigm to hierarchy.

5.2.1 The Stitching Problem and Conditional Sequence Modeling

A major limitation of standard DTs is their reliance on the quality of the training dataset. If the dataset contains only suboptimal trajectories, the DT learns to be suboptimal. It struggles to "stitch" together the good parts of a bad trajectory with the good parts of another to form an optimal path.

DT-HRL addresses this with a two-tiered architecture:

1. **High-Level (MTGC-DT):** A Multi-Task Goal-Conditioned Decision Transformer. Instead of outputting actions, it outputs a sequence of **subgoal embeddings**. It is conditioned on the desired *future return* (reward).
2. **Low-Level (Action Primitives):** A library of parameterized primitives that execute the subgoals.

By predicting subgoals rather than actions, the High-Level DT effectively acts as a "prompt engineer" for the low-level policy. It can look at the offline dataset and learn to sequence subgoals that lead to high rewards, even if those specific sequences never appeared contiguously in the training data. This "hierarchical stitching" capability allows DT-HRL to achieve **>10% higher success rates** and **>8% higher average rewards** in long-sequence manipulation tasks compared to flat Decision Transformers.[27]

---

6. The Frontier of Skill Discovery: Unsupervised and

Disentangled

A persistent bottleneck in HRL is defining *what* the subgoals should be. Hand-crafting them is unscalable. Unsupervised Skill Discovery (USD) aims to learn these subgoals automatically from interaction. The focus in 2025 is on **Disentanglement** and **Density differentiation**.

6.1 Disentangled Unsupervised Skill Discovery (DUSDi)

Previous skill discovery methods (like DIAYN) often learned "entangled" skills—a skill might move the agent forward but also inadvertently spin it, making it hard to combine with other skills.

DUSDi (Disentangled Unsupervised Skill Discovery) [29] introduces a rigorous mathematical framework for learning atomic, reusable behaviors.

- **Mutual Information Objective:** DUSDi maximizes the mutual information between a skill variable z and the state transition, but with a constraint: each component of the skill vector z_i is forced to influence only a specific subspace of the environment (e.g., z_1 affects X-position, z_2 affects Y-position).
- **Value Factorization:** To optimize this, DUSDi utilizes value factorization techniques that allow the critic to estimate the value of these disentangled components separately.

Impact: This disentanglement allows for **Concurrent Composition**. The high-level planner can activate multiple skills simultaneously (e.g., "Run" + "Jump") with predictable results, significantly outperforming entangled baselines in downstream hierarchical tasks.[29]

6.2 Density-Based Skill Differentiation (SD3)

SD3 (State Density Deviation of Different Skills) [30] takes a topological approach to skill discovery.

- **Density Deviation:** Instead of just maximizing entropy (which encourages randomness), SD3 maximizes the *deviation* between the state density of the current skill and the state densities of *all other skills*. This forces the agent to carve out distinct, non-overlapping niches in the state space.
- **Soft Modularization:** SD3 uses a Conditional Variational Autoencoder (CVAE) with a "routing network" to estimate these densities in high-dimensional spaces (like images).

This approach ensures that the learned skills are not just diverse, but functionally distinct, covering the state space more efficiently than methods based on pure entropy or empowerment.[31]

---

7. Applied Hierarchies: Robotics and Complex Strategy

The theoretical advancements discussed above are being validated in high-stakes application domains, proving that HRL is ready for the real world.

7.1 Robotics: Vision-Instruction Correlation (VICtoR)

In robotic manipulation, goals are often specified via natural language (e.g., "Put the red block in the drawer"). The challenge is designing a reward function that understands both the vision (what the robot sees) and the instruction (what the user wants).

VICtoR (Vision-Instruction Correlation Rewards) [32] introduces a hierarchical reward model for long-horizon tasks.

- **The Deficiency of VLMs:** Standard Vision-Language Models (VLMs) can match an image to text, but they lack "temporal awareness." They struggle to distinguish between "reaching for the drawer" and "opening the drawer" if the visual difference is subtle.
- **Hierarchical Decomposition:** VICtoR breaks the reward modeling into two levels:
    1. **Stage Detector:** A high-level classifier that identifies the current "semantic stage" of the task (e.g., Approach, Grasp, Lift, Place).
    2. **Motion Progress Evaluator:** A low-level scorer that measures progress *within* the current stage.
- **Contrastive Losses:** The model is trained using three specific contrastive losses:
    - L_{tcn} (Time Contrastive): Encourages temporal smoothness.
    - L_{mcn} (Motion Contrastive): Aligns motion dynamics with instructions.
    - L_{lfcn} (Language-Frame Contrastive): Aligns visual states with semantic goals.

Results: VICtoR demonstrates a **43% improvement** in success rates for long-horizon manipulation compared to flat VLM rewards (like LIV or LOReL). This confirms that hierarchical *perception* is a prerequisite for hierarchical *control*.[32]

7.2 Strategy Games: Society of Mind in StarCraft II (HIMA)

StarCraft II remains a grand challenge for AI due to its partial observability and multi-agent dynamics. **HIMA (Hierarchical Imitation Multi-Agent)** [33] applies a "Society of Mind" approach to this domain.

- **Strategic Planner (SP):** This is the meta-controller. It does not issue micro-commands (like "move marine to (x,y)"). Instead, it issues **Orders** based on a global strategic assessment (e.g., "Expand Economy," "Harass Enemy," "All-In Attack").
- **Specialized Agents:** HIMA trains distinct imitation learning agents for each specific strategy. These agents are "experts" in their narrow domain.
- **Temporal Chain-of-Thought (t-CoT):** The Strategic Planner uses a novel reasoning mechanism called t-CoT. It maintains a "strategic memory" of the game state, allowing it to align immediate actions with long-term objectives and adapt to the opponent's moves.
- **TextSCII-All:** The authors introduce a new evaluation environment, TextSCII-All, covering all race matchups (Protoss, Terran, Zerg), unlike previous benchmarks which were limited.

HIMA outperforms both built-in AIs and other LLM-based agents by leveraging this specialization. It proves that in complex systems, a hierarchy of specialized experts (coordinated by a generalist planner) is superior to a single monolithic policy.[33]

7.3 The Limits of Hierarchy: The Minecraft Anomaly

Finally, it is crucial to report the limitations. A significant study in the **2D Minecraft Domain** [36] revealed that HRL is not a magic bullet.

- **The Findings:** In goal-oriented environments, without a-priori knowledge or well-shaped rewards, HRL methods frequently failed to outperform standard (flat) RL.
- **The Reason:** The "Abstraction Gap." HRL relies on the agent discovering "bottom-up action abstractions" (skills) that match the "intrinsic top-down decomposition" of the task. If the learned skills (e.g., random movement) don't map neatly to the logical steps of the task (e.g., mining wood), the hierarchy collapses and adds unnecessary complexity.

This negative result is vital: it justifies the complexity of methods like **GLIDER** (which uses offline data to ensure useful skills) and **PIPER** (which ensures subgoal feasibility). Tabula rasa HRL—learning hierarchy from scratch without priors—remains a brittle and often inefficient endeavor.[36]

---

8. Conclusion and Future Outlook

As we advance through 2025, Deep Hierarchical Reinforcement Learning has evolved from a niche optimization technique into the structural backbone of Agentic AI. The field has moved beyond the simple question of "how to learn options" to the profound challenge of "how to align semantic reasoning with physical control."

Key Takeaways:

1. **Architecture is Destiny:** The success of **CoDA** proves that for long-horizon tasks, the architecture of memory (what to forget) is as important as the learning algorithm.
2. **Feasibility is King:** The shift to **Preference-Based Learning (PIPER/DIPPER)** highlights that the stability of HRL depends on the Manager respecting the Worker's limitations. We cannot command what cannot be executed.
3. **Emergence: HICRA** demonstrates that hierarchy is an inherent property of intelligence, emerging naturally in neural networks when credit assignment is correctly focused.

Future Directions:
We anticipate the next leap will involve End-to-End Hierarchical World Models. Rather than separate modules for Planning and Execution, we expect to see unified Transformer architectures trained on massive multi-modal datasets (video + language + action) that learn to compress time dynamically—effectively learning a "video codec" for behavior. Furthermore,

the Society of Mind paradigm (HIMA) suggests that the future of HRL lies in multi-agent internal markets, where specialized sub-policies bid to solve parts of a problem, orchestrated by a central reasoning core.
Deep HRL is no longer just about solving mazes; it is about building the cognitive architecture for autonomous systems that can reason, plan, and act over days, weeks, and years.

Table 1: Comparative Analysis of High-Level Controller Architectures (2025)

Feature	**CoDA (Context-Decoupled Agent)**	**GLIDER (Grounding LLMs)**	**HICRA (Hierarchy-Aware)**
Core Philosophy	**Information Hiding**: Decouple strategy from execution noise.	**Grounding**: Bridge semantic plans with offline physical skills.	**Emergence**: Hierarchy arises from targeted credit assignment.
High-Level Role	**Planner**: Operates in sparse "Strategic Context."	**Planner**: Generates Chain-of-Thought sub-tasks.	**Strategic Tokens**: High-entropy "fork points" in reasoning.
Low-Level Role	**Executor**: Ephemeral context, flushed after task.	**Controller**: Offline RL policy (task-agnostic).	**Procedural Tokens**: Low-entropy execution steps.
Training Method	**PECO**: Group Relative Policy Optimization (GRPO).	**Offline HRL**: Pre-training on massive interaction datasets.	**HICRA**: Weighted policy gradient on planning tokens.
Key Innovation	Solves "Context Explosion" via ephemeral contexts.	Solves "Hallucination" via physical grounding.	Solves "Credit Assignment" via entropy-based weighting.
Primary Domain	Information Retrieval, Multi-hop	Embodied Agents, Robotics	Mathematical Reasoning, Logic

	QA, Coding.	(ScienceWorld).	Puzzles.

Table 2: Comparison of Non-Stationarity Solutions

Feature	**PIPER (Primitive-Informed Preference)**	**DIPPER (Direct Preference Optimization)**
Objective	Learn a **Reward Model** from preferences.	Optimize **Policy** directly (DPO) from preferences.
Feasibility Mechanism	**Regularization**: Conditions Manager on Worker's Value Function (V_{π_L}).	**Reference Policy**: Anchors optimization to a primitive-aware reference.
Feedback Source	**Primitive-in-the-Loop**: Simulated preferences from sparse env rewards.	**Human/Simulated**: Preference pairs between trajectories.
Stability Solution	**Hindsight Relabeling**: Re-labels failed goals as successes.	**Bi-Level Optimization**: Decouples upper/lower level updates.
Performance	>50% success in Franka Kitchen (vs 0% baseline).	Outperforms PIPER in sample efficiency; robust to noise.

Works cited

1. [Literature Review] CoDA: A Context-Decoupled Hierarchical Agent with Reinforcement Learning - Moonlight, accessed December 22, 2025, https://www.themoonlight.io/review/coda-a-context-decoupled-hierarchical-agent-with-reinforcement-learning
2. Divide and Conquer: Grounding LLMs as Efficient Decision-Making Agents via Offline Hierarchical Reinforcement Learning | OpenReview, accessed December 22, 2025, https://openreview.net/forum?id=pdNtji3ktF¬eId=TnkdScrvgI
3. PIPER: Primitive-Informed Preference-based Hierarchical Reinforcement Learning via Hindsight Relabeling - the University of Bath's research portal, accessed December 22, 2025, https://researchportal.bath.ac.uk/en/publications/piper-primitive-informed-prefer

ence-based-hierarchical-reinforcem/

4. DIPPER: DIRECT PREFERENCE OPTIMIZATION FOR PRIMITIVE-ENABLED HIERARCHICAL REINFORCE - OpenReview, accessed December 22, 2025, https://openreview.net/pdf/5cab3bab53d6c012e548bb882be26d1b72727940.pdf
5. Emergent Hierarchical Reasoning in LLMs through Reinforcement Learning - arXiv, accessed December 22, 2025, https://arxiv.org/html/2509.03646v1
6. [2306.16021] Structure in Deep Reinforcement Learning: A Survey and Open Problems, accessed December 22, 2025, https://arxiv.org/abs/2306.16021
7. [2411.18892] A Comprehensive Survey of Reinforcement Learning: From Algorithms to Practical Challenges - arXiv, accessed December 22, 2025, https://arxiv.org/abs/2411.18892
8. Non-stationarity in HRL and its effects on HRL training. - ResearchGate, accessed December 22, 2025, https://www.researchgate.net/figure/Non-stationarity-in-HRL-and-its-effects-on-HRL-training_fig1_382677991
9. (PDF) CoDA: A Context-Decoupled Hierarchical Agent with Reinforcement Learning, accessed December 22, 2025, https://www.researchgate.net/publication/398720105_CoDA_A_Context-Decoupled_Hierarchical_Agent_with_Reinforcement_Learning
10. CoDA: A Context-Decoupled Hierarchical Agent with Reinforcement Learning - arXiv, accessed December 22, 2025, https://arxiv.org/html/2512.12716v1
11. CoDA: A Context-Decoupled Hierarchical Agent with Reinforcement Learning - alphaXiv, accessed December 22, 2025, https://www.alphaxiv.org/resources/2512.12716
12. CoDA: A Context-Decoupled Hierarchical Agent with Reinforcement Learning - 心流, accessed December 22, 2025, https://www.iflow.cn/static/chat?q=CoDA%3A%20A%20Context-Decoupled%20Hierarchical%20Agent%20with%20Reinforcement%20Learning
13. Emergent Hierarchical Reasoning in LLMs through Reinforcement Learning - OpenReview, accessed December 22, 2025, https://openreview.net/forum?id=NlkykTqAld
14. Emergent Hierarchical Reasoning in LLMs through Reinforcement Learning, accessed December 22, 2025, https://tiger-ai-lab.github.io/Hierarchical-Reasoner/
15. Emergent Hierarchical Reasoning in LLMs through Reinforcement Learning - alphaXiv, accessed December 22, 2025, https://www.alphaxiv.org/overview/2509.03646
16. Direct Preference Optimization for Primitive-Enabled Hierarchical Reinforcement Learning, accessed December 22, 2025, https://arxiv.org/html/2411.00361v3
17. PIPER: Primitive-Informed Preference-based Hierarchical Reinforcement Learning via Hindsight Relabeling, accessed December 22, 2025, https://proceedings.mlr.press/v235/singh24e.html
18. PIPER: Primitive-Informed Preference-based Hierarchical Reinforcement Learning via Hindsight Relabeling - GitHub, accessed December 22, 2025, https://raw.githubusercontent.com/mlresearch/v235/main/assets/singh24e/singh24e.pdf

19. [Quick Review] PIPER: Primitive-Informed Preference-based Hierarchical Reinforcement Learning via Hindsight Relabeling - Liner, accessed December 22, 2025, https://liner.com/review/piper-primitiveinformed-preferencebased-hierarchical-reinforcement-learning-via-hindsight-relabeling
20. DIPPER: Direct Preference Optimization for Primitive-Enabled Hierarchical Reinforcement Learning | OpenReview, accessed December 22, 2025, https://openreview.net/forum?id=mJKhn7Ey4y
21. DIPPER: Direct Preference Optimization to Accelerate Primitive-Enabled Hierarchical Reinforcement Learning - arXiv, accessed December 22, 2025, https://arxiv.org/html/2406.10892v1
22. (PDF) DIPPER: Direct Preference Optimization to Accelerate Primitive-Enabled Hierarchical Reinforcement Learning - ResearchGate, accessed December 22, 2025, https://www.researchgate.net/publication/381485319_DIPPER_Direct_Preference_Optimization_to_Accelerate_Primitive-Enabled_Hierarchical_Reinforcement_Learning
23. Hierarchical Message-Passing Policies for Multi-Agent Reinforcement Learning - arXiv, accessed December 22, 2025, https://arxiv.org/html/2507.23604v1
24. [PDF] Feudal Graph Reinforcement Learning - Semantic Scholar, accessed December 22, 2025, https://www.semanticscholar.org/paper/Feudal-Graph-Reinforcement-Learning-Marzi-Khehra/d9854df0a7474414958dd7d45fbfca4cb0f4a4eb
25. Feudal Graph Reinforcement Learning - IRIS Re.Public@polimi.it, accessed December 22, 2025, https://re.public.polimi.it/retrieve/c920f960-fa2b-4617-a11a-0728a0aa556c/3451_Feudal_Graph_Reinforcemen.pdf
26. (PDF) Feudal Graph Reinforcement Learning - ResearchGate, accessed December 22, 2025, https://www.researchgate.net/publication/369946245_Feudal_Graph_Reinforcement_Learning
27. DT-HRL: Mastering Long-Sequence Manipulation with Reimagined ..., accessed December 22, 2025, https://pmc.ncbi.nlm.nih.gov/articles/PMC12467737/
28. DT-HRL: Mastering Long-Sequence Manipulation with Reimagined Hierarchical Reinforcement Learning - ResearchGate, accessed December 22, 2025, https://www.researchgate.net/publication/395218310_DT-HRL_Mastering_Long-Sequence_Manipulation_with_Reimagined_Hierarchical_Reinforcement_Learning
29. Disentangled Unsupervised Skill Discovery for Efficient Hierarchical ..., accessed December 22, 2025, https://proceedings.neurips.cc/paper_files/paper/2024/hash/8c263f70550cc7d69dba3fc170a23e77-Abstract-Conference.html
30. Unsupervised Reinforcement Learning by Maximizing Skill Density ..., accessed December 22, 2025, https://openreview.net/forum?id=RKB4WiesB4
31. [2506.14420] Unsupervised Skill Discovery through Skill Regions Differentiation - arXiv, accessed December 22, 2025, https://arxiv.org/abs/2506.14420

32. VICtoR: Learning Hierarchical Vision-Instruction Correlation ..., accessed December 22, 2025, https://openreview.net/forum?id=UpQLu9bzAR
33. Society of Mind Meets Real-Time Strategy: A Hierarchical Multi-Agent Framework for Strategic Reasoning - ChatPaper, accessed December 22, 2025, https://chatpaper.com/paper/178760
34. Society of Mind Meets Real-Time Strategy: A Hierarchical Multi-Agent Framework for Strategic Reasoning - OpenReview, accessed December 22, 2025, https://openreview.net/pdf/7a9859d0ac848995814c41ceea8ac21116aad9a7.pdf
35. Society of Mind Meets Real-Time Strategy: A Hierarchical Multi-Agent Framework for Strategic Reasoning - arXiv, accessed December 22, 2025, https://arxiv.org/html/2508.06042v1
36. NeurIPS Does Hierarchical Reinforcement Learning Outperform ..., accessed December 22, 2025, https://neurips.cc/virtual/2023/78641