International Journal for Multidisciplinary Research (IJFMR)

E-ISSN: 2582-2160 ● Website: www.ijfmr.com ● Email: editor@ijfmr.com

S-AI: A Sparse Artificial Intelligence System Orchestrated by a Hormonal MetaAgent and Context-Aware Specialized Agents

Said Slaoui

LRI, Mohammed V University of Rabat

Abstract

We introduce S-AI, a Sparse Artificial Intelligence architecture designed for modular, adaptive, and resource-efficient problem solving. At the heart of S-AI lies the Hormonal MetaAgent (HMA), a biologically inspired orchestrator that dynamically coordinates the execution of specialized agents based on an evolving internal context.

The system incorporates an artificial signaling mechanism inspired by endocrine regulation, enabling it to respond to fluctuating demands such as urgency, system load, or contextual trust. Signals such as urgency or high_load influence agent activation modes, triggering simplified or deferred execution when needed.

This adaptive control layer is supported by a decomposition module, which breaks down complex problems into smaller subproblems, and by a suite of domain-specific agents. The combination of sparse activation, targeted orchestration, and hormone-like regulation allows S-AI to reduce computational overhead while maintaining responsiveness.

A dedicated layer of gland agents further enhances this hormonal mechanism by preparing complete behavioral profiles—including context-specific data and modulation parameters—prior to hormone diffusion. This intermediate endocrine-inspired layer enables strategic adaptation without overloading the MetaAgent.

Experimental results show that S-AI achieves substantial gains in efficiency and reactivity without compromising solution quality. The architecture demonstrates how bio-inspired design principles can be leveraged for intelligent coordination in modular AI systems.

Keywords: Sparse Artificial Intelligence (S-AI), Hormonal MetaAgent (HMA), Gland Agents, Artificial Hormonal Signaling, Endocrine-Inspired Coordination, Modular Intelligence Architecture, Adaptive Agent Orchestration, Bio-Inspired Control Mechanisms, Context-Aware Decision Making, Selective Activation, Resource-Efficient AIGeneral Context: Toward Sustainable and Modular AI.

1.1 Introduction

Artificial Intelligence (AI) has achieved remarkable progress over the past decade, driven by advances in deep learning, large-scale data processing, and high-performance computing. From natural language processing to image recognition and autonomous systems, AI is transforming nearly every sector of human activity. However, this growth has come at a cost. Modern AI systems are becoming increasingly monolithic, resource-intensive, and opaque—often requiring massive datasets, heavy computation, and complex end-to-end architectures that are difficult to interpret and control.

This trend raises crucial concerns not only about scalability and energy consumption, but also about maintainability, modularity, and adaptability. The more complex and over-engineered an AI pipeline becomes, the harder it is to adapt it to new tasks, domains, or constraints. In addition, the environmental footprint of large AI models is becoming a growing issue in both academic and industrial contexts.

In this context, there is a growing need for new architectural paradigms that promote modularity, explainability, and energy parsimony, while still achieving intelligent behavior. Rather than continuing to build ever-larger and more complex models, a shift toward lightweight, distributed, and orchestrated intelligence is emerging as a promising alternative. Sparse Artificial Intelligence (S-AI) aims to answer this call by offering a novel approach based on modular agents, selective activation, and dynamic task decomposition.

Beyond its modular and parsimonious foundations, the S-AI architecture is also inspired by biological systems, particularly the human endocrine system. In this perspective, we introduce the concept of artificial hormonal signaling as a complementary mechanism to centralized orchestration. Just as hormones modulate the activity of organs in a decentralized and context-sensitive manner, virtual hormones can diffuse signals across specialized agents in S-AI, influencing their activation, sensitivity, or priority without direct control. This bio-inspired layer enriches the flexibility and adaptability of the system while preserving its modular integrity.

To support this hormonal mechanism, S-AI also incorporates a dedicated layer of gland agents. These intermediary modules are responsible for preparing complete behavioral profiles—including the relevant data and modulation parameters—before hormonal diffusion. Acting as functional analogs of biological glands, they relieve the MetaAgent from the burden of context-specific configuration and enable a more scalable, profile-driven orchestration strategy.

1.2 Motivations and Limitations of Current Deep AI Systems

Despite their undeniable success, current AI systems—especially those based on deep learning—suffer from several structural limitations that hinder their long-term scalability and adaptability. These systems often rely on large end-to-end architectures that require extensive training, vast amounts of annotated data, and significant computational resources. While such approaches perform well in controlled environments, they tend to be rigid, difficult to reconfigure, and costly to maintain in real-world applications.

Moreover, these architectures typically lack modularity: once trained, the system behaves as a black box, making it challenging to isolate components, update specific modules, or understand internal decision processes. This lack of transparency raises concerns in critical domains such as healthcare, law, or finance, where explainability is not optional but a necessity.

Another major drawback is the growing computational and environmental cost of training and deploying large models. The carbon footprint of state-of-the-art models is now being actively studied and criticized, pushing the community to seek more sustainable alternatives.

Sparse AI arises from this pressing need: it proposes a new way to design intelligent systems from the ground up, based on the principles of decomposition, selectivity, modularity, and intelligent orchestration, rather than brute-force training and uniform activation.

1.3 Contributions of the Paper

This paper introduces a complete and modular architecture for Sparse Artificial Intelligence (S-AI), built upon the selective activation of specialized agents, adaptive task decomposition, intelligent orchestration of resources, and bio-inspired regulation mechanisms. The main contributions of this work are as follows:

- A new architectural vision of artificial intelligence, based on the distributed collaboration of lightweight, interoperable, and specialized intelligent agents;
- The definition of a structured repository of Specialized Agents, optimized to efficiently solve targeted sub-problems across multiple domains, and selectively activated according to computational parsimony principles;
- The introduction of a Decomposition Agent, acting as the central engine for adaptive task segmentation, combining symbolic approaches with neuro-symbolic hybridization;
- The design of a MetaAgent, serving as a central orchestrator capable of dynamically selecting the most relevant agents based on context, with feedback-driven learning capabilities;
- The integration of an artificial hormonal signaling mechanism that enables dynamic, context-aware modulation of agent behavior. This includes:
  - The introduction of Gland Agents, functioning as endocrine modulators capable of releasing hormone-like signals in response to internal or external stimuli;
  - The implementation of a Hormonal Engine, responsible for the generation, propagation, and decay of artificial hormones;
  - The construction of a shared Hormonal Context, which acts as a continuous stream of soft signals used to adjust agent priorities, activation thresholds, and coordination strategies in real-time;
- The implementation of a Result Aggregator, enabling coherent, weighted, and explainable fusion of partial results;
- The presentation of a modular functional prototype, demonstrating the practical feasibility of the S-AI paradigm through several representative use cases;
- A thorough discussion of the expected benefits in terms of computational efficiency, traceability, flexibility, and explainability, compared to traditional monolithic AI pipelines.

These contributions aim to demonstrate that it is possible to design intelligent systems that are more efficient, interpretable, and adaptive—without compromising performance—by relying not on brute complexity, but on orchestrated, specialized, and parsimonious intelligence, augmented by a biologically inspired layer ofhormonal regulation.

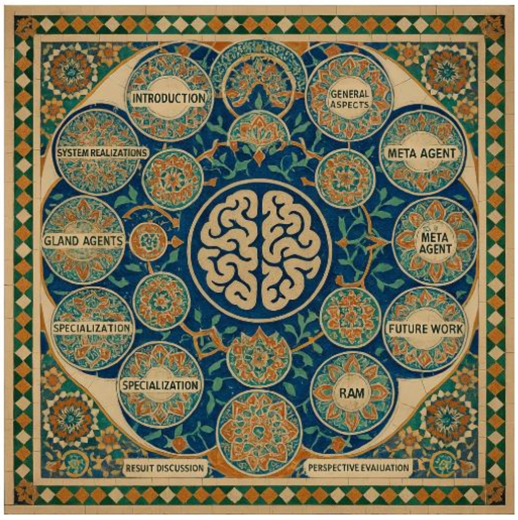


Figure 1.1. Moroccan mosaic overview of the Sparse Artificial Intelligence (S-AI) system.

This visual metaphor, inspired by traditional Moroccan zellige and Islamic geometric art, offers an intuitive and structured representation of the S-AI architecture. At the center of the mosaic lies a stylized brain, symbolizing the Hormonal MetaAgent (HMA), which coordinates the system's overall behavior through adaptive signaling. Surrounding the central core are twelve decorated medallions, each corresponding to a key module of the system—such as Specialized Agents, the Decomposition Agent, the Hormonal Engine, the Result Aggregator, and the Result Access Module (RAM). One newly added medallion is dedicated to the Gland Agents, represented through a spiral or fluid motif, highlighting their role as endocrine-inspired modulators of context-aware behavior. The figure’s circular layout and ornamental borders emphasize the modularity, cooperation, and bio-inspired harmony of the system. Rather than suggesting a linear pipeline, this configuration illustrates a distributed orchestration where knowledge and decisions circulate adaptively among agents.

Section 2 – Theoretical Foundations of Sparse Artificial Intelligence

2.1 Principle of Activation Parsimony

One of the core theoretical pillars of Sparse Artificial Intelligence (S-AI) is the principle of activation parsimony. Unlike traditional AI systems that often activate all components of a model regardless of the specific needs of a problem, S-AI promotes a selective, task-dependent activation of computational resources. This principle draws inspiration from biological systems, where only relevant cognitive processes are mobilized according to context.

In practice, this means that only the minimal subset of agents or modules required for a given subproblem is activated, reducing computational load and increasing interpretability. Activation parsimony aims not only to minimize energy consumption, but also to make system behavior more transparent and explainable. It shifts the design logic from “all-purpose processing” to “intelligent activation,” redefining how resources are managed in AI systems.

2.2 Task Decomposition and Modular Intelligence

S-AI is built on the concept that a complex task can often be decomposed into simpler subproblems, each solvable by specialized agents using lightweight methods. This decomposition-oriented view echoes classical problem-solving techniques but is integrated here as a structural principle of system architecture rather than a heuristic.

By associating each agent with a specific capability (e.g., numeric reasoning, language summarization, image tagging), S-AI forms a network of domain-specific micro-experts, coordinated by a central orchestrator. This modular intelligence architecture increases system flexibility, allows easier updating of components, and promotes reusability across tasks.

2.3 The 20/80 Heuristic in Cognitive and Computational Processing

The S-AI paradigm adopts a 20/80 decomposition heuristic, inspired by the Pareto principle and cognitive ergonomics. The underlying assumption is that 80% of a given problem can often be addressed through lightweight, low-cost methods, while the remaining 20% may require deeper, more complex treatment.

This principle serves as a practical guideline for system design: delegate the majority of routine or predictable subtasks to low-cost agents, and reserve high-cost computational tools (e.g., deep models) for exceptional or ambiguous cases. It enables an architecture that is both scalable and frugal, without sacrificing robustness.

2.4 From Monolithic Pipelines to Smart Orchestration

In contrast to traditional monolithic AI pipelines, where a single end-to-end model handles all steps indiscriminately, S-AI proposes a decentralized architecture with dynamic orchestration. The MetaAgent plays a pivotal role by selecting which agents to activate, adapting the strategy based on task complexity, agent performance history, and contextual feedback.

This smart orchestration not only reduces wasteful computation but also enables context-aware modular execution. The result is a system that behaves more like a network of collaborating cognitive agents than a static processing chain. Such a paradigm opens new possibilities for explainable, scalable, and sustainable artificial intelligence.

2.5 Endocrine Bio-Inspiration and Artificial Hormonal Signaling

A complementary theoretical foundation of S-AI draws inspiration from the human endocrine system, regarded as a model of decentralized and context-sensitive orchestration. This biological system operates through the production and circulation of hormones, which modulate the behavior of distant target organs without relying on direct neural connections. Translating this principle into the realm of sparse artificial intelligence, we introduce the concept of artificial hormonal signaling, aimed at enriching the global coordination model. In S-AI, this signaling mechanism takes the form of virtual contextual messengers that diffuse across the architecture to dynamically modulate the sensitivity, activation, or execution priority of specialized agents. Rather than replacing the MetaAgent, this layer complements its decisions by enabling a form of distributed self-regulation, which facilitates fine-grained adaptation to complex or evolving problem contexts. Artificial hormonal signaling thus provides an additional layer of systemic intelligence, built on indirect, low-coupling, and expressive communication, fully aligned with the core principles of sparsity, modularity, and robustness at the heart of S-AI.

To structure and contextualize this hormonal mechanism, S-AI incorporates a dedicated layer of gland agents, analogous to biological glands. Each gland agent is responsible for preparing a behavioral profile tailored to a specific situation, including relevant data, parameters, and modulation settings. Upon request from the MetaAgent, a gland agent generates the appropriate hormone and orchestrates its diffusion through the system. This architecture enables a scalable and profile-driven form of adaptive orchestration, where the MetaAgent delegates the operational encoding of context to intermediary functional modules—thus reinforcing the biological coherence and systemic parsimony of the overall model.

2.6 Positioning with Respect to Related Work

Sparse Artificial Intelligence (S-AI) builds upon several research directions that aim to overcome the limitations of traditional AI systems regarding scalability, computational efficiency, and interpretability. It synthesizes and extends concepts from modular architectures, the Mixture of Experts (MoE) paradigm, sparse neural networks, cognitive architectures, and task decomposition methods, while introducing a novel orchestration mechanism based on selective activation and agent-based decomposition.

Firstly, modular architectures, such as expert systems or hybrid symbolic/subsymbolic models, promote reusability, specialization, and scalability. S-AI builds upon these foundations by introducing a system-level orchestration via a Decomposition Agent and a MetaAgent, which dynamically allocate subtasks to appropriate specialized agents. This structure is consistent with recent developments in modular multi-agent frameworks such as MASAI [2].

Secondly, the Mixture of Experts (MoE) paradigm has inspired various selective activation mechanisms in neural network architectures. S-AI distinguishes itself by introducing a transparent, interpretable and context-driven agent selection process, reinforced by feedback loops based on performance monitoring. The recent SMoA model [7][8] is a prime example, combining adaptive expert routing with early stopping strategies. Moreover, pruning strategies in MoE systems [9] help optimize resource allocation by deactivating low-impact agents, in line with S-AI's selective philosophy.

Thirdly, while sparsity techniques such as pruning or low-rank approximations are often applied at the model level, S-AI generalizes this concept to the system level, activating only necessary components for a given task. Similar ideas are found in adaptive sparse communication graphs in multi-agent systems [24], and in hierarchical expert gating [11].

In addition to structural sparsity, S-AI incorporates a novel bio-inspired hormonal signaling layer that adds a decentralized and context-sensitive dimension to agent coordination. This mechanism relies on Gland Agents, which emit artificial hormonal signals in response to environmental or internal stimuli. These signals are managed and propagated by a dedicated Hormonal Engine, which simulates hormone dynamics such as release, diffusion, and decay. The resulting Hormonal Context provides a continuous modulation channel that influences the activation thresholds and behavior of specialized agents. Unlike conventional control signals, this layer enables smooth, indirect regulation of agent interaction and priority, providing a flexible alternative to hard-coded orchestration.

Fourthly, cognitive architectures such as SOAR and ACT-R have shown the potential of modular reasoning systems. S-AI, however, adopts a more pragmatic decomposition strategy focused on operational efficiency rather than full cognitive emulation. This is reflected in hybrid systems such as DARA [3], DIEM [4], and Agentic Retrieval-Augmented Generation for Time Series Analysis [13], which combine semantic decomposition with symbolic reasoning.

Fifthly, task decomposition has long been a core topic in distributed multi-agent systems [5][6], especially for cooperative planning. S-AI innovates by proposing a generic and reusable Decomposition Agent, combined with intelligent agent activation and result aggregation governed by a MetaAgent. Recent frameworks such as TDAG [25] and Parametric Subtasks decomposition in meta-reinforcement learning [12] reflect similar principles.

Furthermore, S-AI's theoretical foundations align with classical AI frameworks that integrate symbolic reasoning, learning, and hierarchical control [14].

In addition to synthesizing existing modular and sparse paradigms, S-AI distinguishes itself through the introduction of biologically inspired mechanisms, implemented with a computational and interpretable perspective. This orientation opens new avenues for adaptive, low-overhead coordination in next-generation AI architectures.

Finally, recent works by Slaoui et al. [15–23] highlight the importance of parallel computing, clustering and scalable architectures, complementing the S-AI approach. Additionally, broader perspectives such as AI Agents evolution [1], orthogonal experts in multi-task RL [11], vision-language MoE adapters [26], and neural decomposition strategies [10] reinforce the relevance of modular, efficient, and agent-based AI design paradigms.

Section 3 – General Architecture of the S-AI System

3.1 – Modular Agent-Based Structure

The architecture of the S-AI system is built upon a modular structure composed of several types of cooperating Specialized agents. Each agent is responsible for a specific function within the overall processing pipeline, enabling an intelligent distribution of tasks, parsimonious resource activation, and natural system scalability.
Among these components, specialized agents play a central role: they constitute the primary execution units responsible for solving the majority of simple and well-defined subtasks. These specialized agents are designed to intervene in a fast, targeted, and resource-efficient manner, in full alignment with the activation parsimony principle that underpins S-AI. They are dynamically activated by the MetaAgent, based on the outcome of the decomposition process.
The system also includes:
a Decomposition Agent, in charge of breaking down complex problems into elementary subtasks;
a MetaAgent, the central orchestrator responsible for supervising the activation of both specialized and expert agents;
expert agents, which are only mobilized for complex cases when specialized agents are insufficient;
a Result Aggregator, responsible for merging and consolidating the outputs from the various agents;
and a Result Accessing Module (RAM), which interfaces the internal system outputs with the external user layer.
This modular architecture enables the construction of a distributed, parsimonious, and adaptive artificial intelligence system, grounded in the interoperability of specialized agents.

3.2 – Specialized Agents (SAs)

Specialized agents constitute the core execution units of the S-AI system. Each agent is designed to handle a specific class of sub-problems based on domain knowledge, computational strategies, or task typology. These agents are lightweight and modular, capable of being activated selectively by the MetaAgent. While this section provides a general architectural overview, a detailed description of their internal mechanisms, specialization strategies, and domain-specific configurations is presented in Section 4.

3.3 MetaAgent: Architectural Role and System Governance

The MetaAgent occupies a central position in the architecture of the S-AI system. It acts as the system's cognitive orchestrator, ensuring the strategic coordination of specialized agents, specialized expert agents, and the decomposition process, which it supervises upstream. The MetaAgent initiates the task processing workflow, determines whether decomposition is required, oversees task allocation, and optimizes agent activation based on resource efficiency and past performance data. It thus serves as the key driver of activation parsimony and system-wide coherence. Its close interaction with the Decomposition Agent enables a hierarchical and modular management of complex tasks.
The detailed mechanisms of agent selection, adaptive learning, performance feedback, and orchestration strategies implemented by the MetaAgent are described in depth in Section 6.

3.4 – Gland Agents: Endocrine Modulators for Contextual Adaptation

As part of its biologically inspired architecture, S-AI includes a dedicated layer of gland agents, which play a crucial role in the artificial hormonal signaling mechanism (see Section 2.5). These agents are designed to act as intermediary modules between the MetaAgent and the Hormonal Engine. They are responsible for preparing full behavioral profiles corresponding to specific system-level configurations,

such as modes of operation (e.g., exploration, resilience, or energy-saving), contextual constraints, or adaptive execution parameters.
Upon receiving a strategic intent from the MetaAgent—such as the activation of a particular behavior profile—the relevant gland agent gathers the appropriate data, computes modulation parameters, and generates an enriched virtual hormone tailored to the current context. This hormone is then diffused system-wide and interpreted by Specialized Agents that are sensitive to its signal. The gland agent thus enables profile-driven orchestration, relieving the MetaAgent from low-level configuration tasks and allowing for more scalable and modular decision-making.
Gland agents are structurally analogous to their biological counterparts: they do not perform computation directly but instead act as context encoders and hormone synthesizers, preparing and diffusing signals that shape the collective behavior of the system. Their introduction enhances the flexibility of S-AI by supporting indirect, low-coupling regulation mechanisms, while remaining fully compatible with the principles of sparsity, modularity, and parsimony.

3.5 Role of the Decomposition Agent

The Decomposition Agent plays a pivotal role as a cognitive interface within the S-AI architecture. Activated under the supervision of the MetaAgent, it acts as the entry point for intelligent task structuring. When the MetaAgent decides that decomposition is appropriate, it delegates the analysis to the Decomposition Agent, which semantically analyzes the problem and segments it into smaller, well-defined subproblems. These subproblems are categorized by type and passed back to the MetaAgent for strategic allocation to the most appropriate specialized agent(s). This component supports the principle of activation parsimony by enabling modular task segmentation that facilitates minimal and efficient agent activation. The decomposition process may rely on symbolic rule-based engines, learned classification models, or hybrid neuro-symbolic approaches, as described in Section 5.

3.6 Communication and Control Layers

The agents and orchestrators within S-AI interact through a lightweight communication layer, supporting the exchange of inputs, results, and metadata. This layer also includes a control protocol, allowing the MetaAgent to supervise agent behavior, initiate re-routing, or suspend unnecessary computation. By separating computation from control, S-AI reinforces its modularity and opens the door to multi-agent cooperation or distributed deployment, including edge or cloud-based architectures.

3.7 Artificial Hormonal Signaling Layer in the S-AI Architecture

In addition to the centralized orchestration mechanism managed by the MetaAgent, the S-AI architecture integrates an artificial hormonal signaling layer, inspired by the functioning of the human endocrine system. This layer enables the diffusion of virtual signals called “hormones”, which carry contextual information and indirectly influence the behavior of Specialized Agents. Unlike explicit instructions issued by the MetaAgent, these hormones operate transversally and with low coupling, dynamically modulating the sensitivity, availability, or activation priority of agents.
Each hormone is associated with a specific context (e.g., system overload, temporal urgency, topic criticality) and has a defined scope of influence, affecting one or more agents depending on their functional profile. Hormones are generated by the Gland Agents, based on encountered events or the global system

state. They are then propagated through a shared memory or contextual bus, which is consulted in real time by agents during activation or decision-making.
This mechanism introduces a form of adaptive distributed regulation, capable of responding to weak signals without requiring explicit coordination. It provides an additional layer of control that is particularly useful in dynamic, ambiguous, or multi-domain environments, while reinforcing the overall robustness, flexibility, and resource-efficiency of the system.

3.8 Result Aggregation and Feedback Channels

Once the agents return their outputs, a dedicated Result Aggregator module collects, evaluates, and synthesizes the partial results into a coherent global answer. The aggregation strategy depends on the task nature and may involve majority voting, weighted averaging, or sequential composition. A feedback channel ensures that the MetaAgent receives performance metrics, error rates, and confidence levels, which are used to improve future decisions. This closed-loop mechanism reinforces the learning capacity of the orchestration layer, making the system increasingly efficient over time.

3.9 Comparative Positioning with Related Modular and Multi-Agent Architectures

This subsection aims to contextualize the S-AI architecture by comparing it with a set of recent or foundational contributions in modular, multi-agent, or resource-efficient artificial intelligence. The objective is to highlight the distinctive features of S-AI, notably its hierarchical structure, explicit task decomposition, activation parsimony, and cognitive orchestration ensured by a MetaAgent.

3.9.1 Approaches focused on sparsity

In-depth comparative analysis: S-AI vs SMoA
Among existing works on sparse architectures in multi-agent systems, SMoA (Sparse Mixture-of-Agents) [7][8] is often cited as a recent and promising framework. However, a deeper comparison with S-AI reveals major conceptual and architectural differences. SMoA relies solely on sparse activation within a flat set of expert agents, usually large language models (LLMs). A gating mechanism selects a limited subset of agents for each task depending on context. While this approach helps reduce computational costs, it does not offer any explicit task decomposition or intelligent coordination entity.
S-AI, on the contrary, introduces two fundamental architectural innovations that are completely absent from SMoA: The MetaAgent: In S-AI, the MetaAgent acts as a central coordinator responsible for the adaptive selection of agents, task distribution, and performance tracking. Unlike the static gating function of SMoA, the MetaAgent enables dynamic contextual orchestration, integrating feedback loops, agent scoring, and co-evolutionary strategies.
The Decomposition Agent: S-AI includes a dedicated agent for semantic task segmentation, enabling structured preprocessing before agent activation. SMoA offers no such mechanism.
The 20/80 heuristic: S-AI relies on a heuristic inspired by the Pareto principle, whereby 80% of subtasks can be processed by simple and lightweight agents, while only 20% require high-computational power. This strategic resource allocation logic is absent from SMoA.
Moreover, S-AI introduces a novel layer of modulation inspired by the endocrine system, through the use of Gland Agents, which simulate the behavior of biological glands by releasing artificial hormonal signals in response to contextual stimuli. These signals are regulated by a dedicated Hormonal Engine, which

manages their synthesis, diffusion, and decay over time, creating a hormonal context that influences agent activation, prioritization, and resource allocation.
This hormonal signaling system adds an additional layer of contextual adaptivity, allowing S-AI to dynamically regulate the intensity and focus of processing throughout the architecture — a feature entirely absent from SMoA's flat and stateless coordination paradigm.

Other Compared Approaches:
MoE Pruning – A Provably Effective Method for Pruning Experts in Fine-tuned Sparse MoE [9]: This work aims to optimize expert selection in Mixture-of-Experts architectures through a dynamic pruning mechanism based on performance. It shares with S-AI the goal of resource-efficient activation, but this sparsity remains confined within the model itself. S-AI extends it at the system level, via contextual coordination and orchestrated activation by the MetaAgent.
S-AI further enhances this optimization by integrating artificial hormonal signals, released by Gland Agents and regulated by a Hormonal Engine, which dynamically influence agent selection based on the global hormonal context.
Vision-Language MoE Adapters [26]: This approach introduces specialized experts for continuous multimodal learning, focusing on efficient adaptation. While aligned with the parsimony principle, it remains specific to one domain (vision-language), whereas S-AI generalizes this principle across all types of tasks.
Moreover, this generalization in S-AI is supported by inter-agent hormonal signaling, enabling the activation of domain-specific profiles based on a hormonal context modulated by Gland Agents.
Orthogonal Experts – Multi-task Reinforcement Learning [11]: The idea of orthogonal specialization aligns with S-AI's targeted activation logic. However, expert selection in S-AI is orchestrated by the MetaAgent and contextualized, whereas here it relies on internal algebraic constraints. This contextualization is further reinforced in S-AI by an artificial endocrine model, where the dynamic hormonal context guides agent selection alongside the MetaAgent's reasoning.
Sparse adaptive communication in MARL systems [24]: This approach aims to reduce communication costs using a sparse interaction graph. S-AI applies parsimony at a more global and cognitive level, beyond simple inter-agent message exchanges. This global level is further enhanced in S-AI by the introduction of a shared hormonal context, orchestrated by Gland Agents and the Hormonal Engine, acting as a latent layer of indirect inter-agent communication.

3.9.2 Approaches focused on decomposition

TDAG – A Multi-Agent Framework based on Dynamic Task Decomposition and Agent Generation [25]: TDAG introduces a dynamic pipeline of task decomposition and agent generation, based on LLMs and adaptive prompt engineering. While the decomposition philosophy is shared, S-AI stands out with its reusable Decomposition Agent and orchestrated hierarchy.
Additionally, S-AI enhances this decomposition process through artificial hormonal modulation: Gland Agents emit signals that influence the internal state of the system, allowing the Decomposition Agent to dynamically adjust task granularity and rule activation according to the evolving Hormonal Context.
Parametric Subtasks – Parameterizing Meta-RL Tasks via Subtask Decomposition [12]: This approach follows a similar logic of task granularity but remains limited to RL and lacks orchestration or aggregation mechanisms.

S-AI complements its orchestration by introducing a hormonally modulated environment, where subtask decomposition is not only semantically grounded but also sensitive to the system's internal physiological signals, enabling a more adaptive parameterization strategy.
Neural Decomposition [10]: This decomposition is applied inside neural networks, while S-AI applies it at the agent level for interpretable modularity.
DIEM – Decomposition-Integration Enhancing Multimodal Insights [4]: A related approach, but it lacks an explicit decomposition agent and orchestration mechanism such as S-AI's MetaAgent. Furthermore, DIEM does not benefit from the hormonal signaling layer present in S-AI, which allows decomposition strategies to be dynamically modulated in response to contextual signals generated by the Hormonal Engine.
Agentic RAG – Time Series Analysis [13]: Although structured, this system includes neither formal task decomposition nor adaptive agent activation as in S-AI.
S-AI's originality also lies in its neuro-symbolic decomposition process modulated by endocrine-inspired signals, enabling it to adapt both the structure and temporal resolution of subtasks — a crucial advantage in dynamic environments like time series analysis.

3.9.3 Approaches focused on specialized agents and multi-agent coordination

MASAI – Modular Architecture for Software-engineering AI Agents [2]: MASAI proposes a structure of specialized agents coordinated by a scheduler, but without a hierarchical MetaAgent or decomposition logic. S-AI goes further by integrating both components.
In addition, agent coordination in S-AI is reinforced by an artificial hormonal signaling layer, where Gland Agents dynamically influence the selection and reactivity of specialized agents based on contextual needs.
DARA – Decomposition-Alignment-Reasoning Agent [3]: DARA combines different modules for knowledge graph-based question answering. S-AI shares this modularity but expands it through cognitive orchestration and a generalized decomposition agent.
This orchestration is also supported in S-AI by an internal hormonal context managed by the Hormonal Engine, enabling smooth adaptation of agent activation in response to the system's cognitive dynamics.
TDAG [25] – already discussed under decomposition, can also be categorized here due to its agent generation logic, although it lacks a hierarchical design.
S-AI complements this logic by introducing hormonal governance of the agent lifecycle, where artificial endocrine signals influence real-time agent activation, deactivation, and adaptation.
AI Agents – Evolution, Architecture and Real-World Applications [1]: A conceptual overview of multi-agent architectures. S-AI operationalizes this vision in a functional system. This system notably relies on implicit communication via hormonal signals, enriching inter-agent interactions without increasing explicit messaging overhead.
Foundational works: Russell & Norvig [14], Durfee [5], Lesser [6], Wooldridge [27]: S-AI can be viewed as a modern and parsimonious implementation of the principles presented in these foundational works. It updates these principles through the integration of an artificial endocrine model, providing smooth, continuous, and distributed regulation of multi-agent activation.

3.9.4 Bio-Inspired Approaches and Indirect Behavior Modulation

While most modular and multi-agent AI architectures focus on explicit coordination or agent activation mechanisms, very few explore indirect modulation strategies inspired by biological systems. The

integration within S-AI of an artificial hormonal signaling system, explicitly modeled after the human endocrine system, represents a novel contribution in this regard.
This mechanism is implemented through dedicated Gland Agents and a centralized Hormonal Engine that jointly manage the release, diffusion, and temporal decay of artificial hormones within the system.
This mechanism enables the diffusion of non-binding contextual signals that softly and asynchronously influence the behavior of specialized agents in a decentralized manner.
These signals form a dynamic Hormonal Context that acts as a latent layer of influence across the agent ecosystem, promoting adaptive behavior without requiring explicit commands.
To the best of our knowledge, no existing architecture incorporates such a dedicated layer of diffuse, asynchronous, low-coupling regulation at the level of a cognitive artificial system. Some works on local feedback loops or adaptive communication in Multi-Agent Reinforcement Learning (MARL) systems, such as those by Foerster et al. [28], Zhang et al. [29], and Kim et al. [30], exhibit partial similarities, but without a dedicated signaling layer or explicit endocrine-inspired design.
In contrast, S-AI's hormonal subsystem is structurally embedded within the architecture and operates in synergy with the system's cognitive agents.
The artificial hormonal signaling mechanism in S-AI enables a form of dynamic self-regulation that complements the centralized orchestration performed by the MetaAgent, thereby enhancing the system's robustness, adaptability, and computational frugality.
Section 4 – Specialized Agents (SAs)

4.1 Concept and Functional Role of Specialized Agents (English version)

A specialized agent is an autonomous software entity designed to perform a well-defined task within a specific domain. It operates based on predefined rules, heuristics, or optimized algorithms, allowing it to solve problems efficiently without requiring self-learning or dynamic adaptation. Unlike an expert agent, a specialized agent does not alter its behavior in response to context; instead, it applies a standardized and proven solution to typical cases.
In the architecture of Sparse Artificial Intelligence (S-AI), specialized agents represent the core execution units. Each agent is tailored to handle a narrow class of sub-problems, such as numerical reasoning, image classification, symbolic analysis, or clinical data interpretation. These agents are selectively activated by the MetaAgent depending on the nature, domain, and complexity of the sub-task identified by the Decomposition Agent.
Thanks to their modular design and low computational footprint, specialized agents enhance the reusability, scalability, and maintainability of the overall system. Their activation follows the principle of parsimony: only the agents that are strictly necessary are engaged, thus optimizing resource usage and improving interpretability. Their functional independence also ensures seamless integration within the S-AI framework, while maintaining a high level of task-specific operational specialization.
Some specialized agents exhibit cross-domain functional reusability, enabling them to operate across different application areas. For instance, an anomaly detection agent could be used both in medicine (for early diagnostic purposes) and in cybersecurity (for intrusion detection). This cross-context adaptability strengthens the modular value of the S-AI framework.
Moreover, specialized agents are not constantly active; they are conditionally and selectively triggered. This on-demand activation, controlled by the MetaAgent, aligns with the core principle of parsimony and clearly differentiates them from traditional monolithic expert agents

4.2 Representative Use Cases and Domain-Specific Specialized Agents

Specialized Agents (SAs) are designed to solve well-defined tasks within specific application domains. Their effectiveness relies on the integration of domain knowledge, targeted techniques, and optimized algorithms. The following examples illustrate the diversity of use cases covered by these agents:

- In medicine, a specialized agent for medical image analysis is responsible for detecting tumors or anomalies in radiographs or MRIs, using compact visual recognition models.
- In finance, a specialized agent for market analysis monitors stock trends and forecasts fluctuations using time-series models and economic heuristics.
- In industrial design, a specialized agent for 3D modeling assists in the automated creation of CAD models based on predefined functional and geometric constraints.
- In cybersecurity, a specialized agent for intrusion detection analyzes network traffic in real time to identify attacks or anomalous behaviors.
- In transportation, a specialized agent for traffic optimization dynamically adjusts routes based on real-time road conditions and unforeseen events.
- In agriculture, a specialized agent for crop monitoring processes satellite imagery to detect early signs of disease or water stress on cultivated plots.

These examples demonstrate that each specialized agent is designed to fulfill a specific function with high efficiency within a clearly defined scope. Despite the diversity of their internal implementations, all agents conform to a shared interface protocol, ensuring seamless integration and coordination within the S-AI architecture.

Each application domain may rely on a structured library of standard specialized agents, forming a catalog of optimized components for recurring tasks. These libraries facilitate reuse, dynamic selection, and operational scalability across the S-AI system.

4.3 Structural and functional distinction between specialized and expert agents

It is essential to distinguish between a specialized agent and an expert agent, as these two concepts are often conflated in the literature. The distinction lies in both their internal structure and functional capabilities.

An expert agent possesses advanced analytical and adaptive abilities. It incorporates techniques such as machine learning, deep learning, or symbolic inference, enabling it to modify its behavior over time based on contextual information and accumulated experience. It can refine its predictions, adapt its decisions continuously, and manage complex or novel situations with a high degree of autonomy. This type of agent is typically more computationally demanding, less predictable, but capable of handling intricate reasoning tasks.

In contrast, a specialized agent is designed to perform a single, well-defined task in a rapid, reliable, and deterministic manner. It relies on fixed rules, heuristics, or optimized algorithms. It delivers a standardized response to common cases with maximum efficiency within a limited scope.

Within the S-AI architecture, specialized agents are prioritized to address the majority of simple and recurring sub-problems (approximately 80%), in accordance with the principle of activation parsimony. Expert agents are activated only in the 20% of cases where the task is too complex to be solved by predefined strategies and requires contextual adaptation or multi-level reasoning. This functional separation reinforces the system's modularity, transparency, and computational frugality.

The distinction between specialized agents, expert systems, and modular AI paradigms (MoE, MAS) can be summarized as follows:

Criterion	Expert Systems	Mixture of Experts (MoE)	Multi-Agent Systems (MAS)	Specialized Agents (S-AI)
Model	Rules / heuristics	Neural network-based experts	Autonomous cooperating agents	Lightweight task-specific modules
Activation	Always active	Weighted (gating mechanism)	Based on internal interaction	Selective via MetaAgent
Optimization	Static	Learned through training	Self-organized	Selective and parsimonious
Objective	Domain-focused resolution	Learning which expert to activate	Agent collaboration	Efficient subtask resolution

This comparison emphasizes the structural novelty of S-AI’s specialized agents, combining modular design, low computational cost, and centralized orchestration by a high-level controller.

4.4 Lifecycle, Adaptation, and Governance of Specialized Agents

The integration of specialized agents within the S-AI system follows a structured and modular lifecycle, designed to ensure long-term system scalability, maintainability, and performance.

Each specialized agent is registered in a centralized yet modular repository, where it is associated with a specific domain of expertise, versioned, and documented according to standardized interface and compatibility specifications. This structured registration facilitates dynamic discovery and reuse. Specialized agents are organized into domain-specific libraries, where each agent is indexed, versioned, and documented according to its technical and functional specifications. These structured libraries improve the discoverability, reuse, and governance of the agent catalog and align with the parsimony-driven orchestration strategy of the MetaAgent.

At runtime, the MetaAgent is responsible for selecting the most appropriate specialized agent for each sub-task, based on domain alignment, historical performance, and availability. If no suitable specialized agent exists, the MetaAgent may escalate the task to an expert agent. This decision-making mechanism ensures the system maintains computational parsimony while remaining responsive to complex scenarios.

Throughout their operational life, specialized agents are subject to continuous performance monitoring. Their outputs are evaluated based on accuracy, latency, and contextual relevance. This historical feedback is used to refine future selection and to inform lifecycle decisions.

As tasks evolve, certain specialized agents may become obsolete, suboptimal, or redundant. The S-AI architecture supports incremental updating, replacement, or merging of agents, ensuring the system remains adaptive without compromising modularity. This allows for the seamless integration of improved techniques or new domain knowledge.

Maintaining a clear distinction between specialized agents and expert agents is essential to preserve the lightweight, interpretable, and resource-conscious philosophy of S-AI. This lifecycle approach ensures targeted problem-solving while reinforcing the long-term sustainability and flexibility of the overall system.

4.5 Inter-Agent Coordination within the S-AI Ecosystem

The effectiveness of Specialized Agents (SAs) depends not only on their individual performance but, above all, on their dynamic coordination with the MetaAgent and the Decomposition Agent. This inter-agent synergy is a fundamental pillar of the S-AI architecture, ensuring a smooth articulation between modularity, adaptability, and efficiency.

When a global problem is submitted to the system, the MetaAgent performs an initial macro-level analysis. It evaluates the nature, domain, complexity, and structure of the problem in order to determine whether it can be directly handled by one or more specialized agents, or whether prior decomposition is required. If decomposition is deemed necessary, the MetaAgent delegates this responsibility to the Decomposition Agent.

The Decomposition Agent then segments the problem into coherent sub-tasks, ensuring that each of them matches an input-output pattern mastered by at least one existing specialized agent. In cases of incompatibility, fallback strategies may be triggered: local restructuring, invocation of an expert agent, or default reasoning mechanisms.

The MetaAgent then takes over again to dynamically select the specialized agents to be activated for each sub-task. This selection is based on criteria such as specialization, historical performance, contextual relevance, and availability. It may evolve in real time as intermediate results are received or as the context changes.

During execution, specialized agents communicate with the MetaAgent through standardized interfaces. The results may be aggregated immediately or refined through iterative feedback loops, depending on the task profile. Several agents may also cooperate in parallel or in sequence, depending on the degree of interdependence between sub-tasks.

This architecture promotes seamless collaboration between autonomous yet orchestrated agents, enhancing the system’s scalability, computational parsimony, and interpretability. It fully embodies the philosophy of S-AI, built upon intelligent orchestration, targeted activation, and resource optimization.

4.6 Learning Capabilities and Evolutionary Strategies of Specialized Agents (English version)

In the S-AI approach, specialized agents are designed to be lightweight, focused, and highly optimized for a specific task. However, this does not mean they are necessarily static or unchangeable over time. The system's modular architecture allows for various forms of evolution, ranging from incremental updates to lightweight supervised learning.

Three main adaptation strategies are considered:

Localized supervised learning: some specialized agents can be enhanced via light adjustment mechanisms, such as fine-tuning parameters based on annotated examples, without altering their overall structure.

Version-controlled updates: agents can be replaced by new versions trained offline or in controlled environments, and validated prior to reintegration. This ensures system stability while allowing for gradual improvement.

MetaAgent-guided adaptation: in certain cases, the MetaAgent may suggest evolving a specialized agent based on repeated feedback, poor performance scores, or identified inconsistencies on borderline cases.

The choice between these mechanisms depends on the application domain, task criticality, and traceability constraints. The overarching goal remains to preserve computational frugality while allowing agents to remain effective in changing contexts.

S-AI does not impose a single learning paradigm for specialized agents but instead provides an open framework to balance stability, specialization, and adaptability. Each agent thus becomes a potentially evolving component, without compromising the system's overall interpretability.

4.7 Embedded Knowledge Base and Functional Autonomy of Specialized Agents

Specialized Agents (SAs) in the S-AI framework rely on an embedded knowledge base, which serves as a localized cognitive core enabling efficient task execution within their specific domain of expertise. This knowledge base (KB-SA) consists of structured resources tailored to the agent's function: inference rules, symbolic models, recognition patterns, pre-indexed data structures, or domain-specific ontologies.

Unlike a centralized shared knowledge base, the embedded KB is local, lightweight, and task-specific. It enables fast execution, functional independence, and reusability of the module across similar use cases. Such autonomy is fully aligned with the parsimony-driven philosophy of S-AI.

Each KB-SA can evolve over time through:

manual injection of new rules or domain cases,

feedback integration from the MetaAgent,

or fusion with other agents that share overlapping functional scopes.

The evolution of an embedded knowledge base follows a controlled stability principle: its structure remains consistent, while incremental additions allow adaptation without requiring deep learning. This knowledge-centric learning strategy reinforces both the traceability and the explainability of each Specialized Agent. This local autonomy can be extended through a more subtle form of context-aware regulation: hormonal reactivity, which is introduced in the following subsection.

4.8 Hormonal Reactivity of Specialized Agents

In addition to their activation by the MetaAgent, Specialized Agents in S-AI integrate a hormonal reactivity capability, allowing them to dynamically adapt their behavior based on contextual signals diffused throughout the system. Each agent possesses a hormonal sensitivity profile, defined by a set of rules, thresholds, or modulation functions that govern how it responds to specific hormones.

For instance, a hormone associated with temporal urgency may lead certain agents to trigger a rapid-execution mode, simplify their internal processing, or reject sub-tasks considered non-essential. Another hormone related to system overload may prompt an agent to delay execution or delegate the task to a less solicited counterpart.

In many cases, the hormones that trigger such contextual adaptations are not directly emitted by the MetaAgent, but rather originate from dedicated gland agents (see Section 3.4). These intermediary modules are responsible for encoding complete behavioral profiles into hormonal signals, including contextual parameters such as urgency thresholds, task criticality, or execution simplification levels. Specialized agents interpret these enriched hormones through their sensitivity profiles, allowing them to fine-tune their behavior in alignment with the system's global state without requiring additional communication. The collaboration between gland agents and hormonal reactivity mechanisms ensures a scalable and distributed form of modulation across the S-AI ecosystem.

This hormonal reactivity offers a complementary mechanism to the explicit orchestration handled by the MetaAgent. It enables agents to make local, context-sensitive decisions without requiring direct coordination, while still maintaining global coherence through the shared hormonal flow. This loosely coupled yet intelligent regulation enhances the adaptivity of the overall S-AI architecture by empowering agents with autonomous, self-modulated behavior aligned with the system's broader operational constraints.

Section 5 – The Decomposition Agent

5.1 Introduction

The architecture of Sparse Artificial Intelligence (S-AI) relies heavily on a key principle: activating only the necessary components for solving a given problem. The efficiency of such a system depends critically on the quality of the initial decomposition process. The component responsible for this step is the Decomposition Agent, which transforms a complex problem into a set of targeted subproblems that can be handled by lightweight specialized agents.

5.2 Semantic Rule-Based Decomposition

The first version of the Decomposition Agent relies on a symbolic engine based on semantic rules. This engine uses a natural language parsing process (NLP) to extract, from a complex user query, the syntactic structure, dominant action verbs, key entities, and relevant semantic cues. These elements are then matched against a set of conditional rules (if/then), encoded as linguistic patterns, to automatically classify each segment of the question and route it to the appropriate specialized agent.

Semantic rules may combine several dimensions:

- The lexical category of terms (verbs, nouns, adjectives)
- The semantic field of concepts (finance, climate, biology, etc.)
- The syntactic relationship between words (verb–complement, noun–attribute, etc.)
- The canonical form of the question (e.g., estimation, comparison, causal impact)

Example of a simple conditional rule:
IF the main verb is "calculate" AND the associated noun is "average"
→ Task type = "Math"
→ Agent to activate = AgentMathsSimple

Illustrative Example: Decomposing the Problem "How Long Can Bitcoin Resist?"

Let us now present a concrete example that illustrates the effectiveness of this semantic rule-based reasoning. The question "How long can Bitcoin resist?" is a typical complex problem, combining multiple implicit dimensions—economic, technological, environmental, and regulatory—which must be decomposed into well-targeted subproblems, each suitable for a specific specialized agent.

From the initial parsing phase, the extracted key concepts include:

- Bitcoin (central object),
- resist (abstract action verb),
- and several implicit concerns: price evolution, volatility, institutional adoption, technical innovation, etc.

Through a cascade of enriched semantic rules, this question is automatically decomposed into 10 subproblems, including 8 light tasks (handled by simple agents) and 2 complex tasks requiring advanced deep learning agents.

Simple subproblems (80%):

1. Historical price analysis → Time series agent (AgentSeries)
2. Current and future volatility → Regression agent (AgentForecasting)
3. Market sentiment → NLP agent (AgentSentiment)

4. Institutional adoption → Statistical agent (AgentFinance)
5. Impact of regulations → Symbolic reasoning agent (AgentLegal)
6. Technological developments → Monitoring agent (AgentTechScan)
7. Competition from other cryptocurrencies → Comparative analysis agent (AgentAltcoins)
8. Environmental factors → Impact calculation agent (AgentEnergyFootprint) Complex subproblems(20%):
9. Forecasting economic crises affecting Bitcoi → Requires LSTM / Transformers (AgentCrisisForecast)
10. Simulating extreme scenarios (regulatory bans, collapses, attacks) →Requires GANs or strategic simulation agent (AgentScenarioSim)

This decomposition clearly demonstrates the capability of the symbolic engine to generate a fine-grained map of sub-tasks to activate, while respecting the core principles of sparse computation: only the last two subproblems require heavy agents. The rest are handled efficiently using lightweight, cost-effective modules.

Such semantic rule-based processing, while symbolic, offers a robust and interpretable foundation for initiating deductive reasoning, prior to adaptive learning mechanisms (see Section 5.3) or neuro-symbolic hybridization (see Section 5.4).

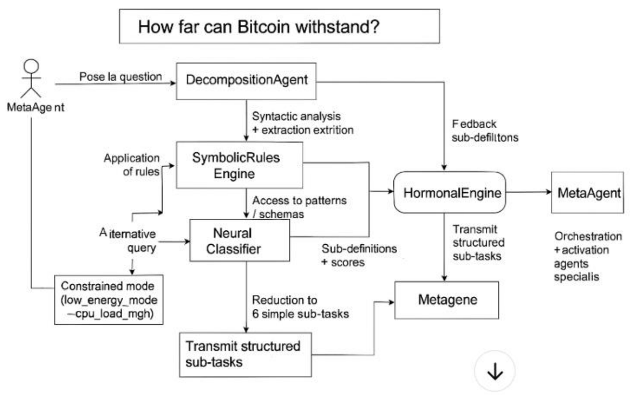


Figure 5.2 – UML diagram of the symbolic decomposition process.

This diagram illustrates the processing flow executed by the Decomposition Agent on a complex query, including semantic parsing, rule-based segmentation, optional activation of the neural classifier, contextual adaptation via hormonal signaling, and structured task transmission to the MetaAgent.

5.3 Rule Learning from Annotated Data

While the symbolic engine initially relies on a handcrafted set of semantic rules (see Section 5.2), it can be further enriched and generalized through a process of supervised learning from annotated data. This adaptive capability allows the Decomposition Agent to evolve its rule base over time as new examples are encountered.

The principle is as follows: given a corpus of complex problems manually annotated with their corresponding decomposed subproblems, the system can automatically extract recurring linguistic, lexical, and semantic patterns. These patterns are then used to generate new decomposition rules, formalized as enriched semantic templates.

Techniques used:

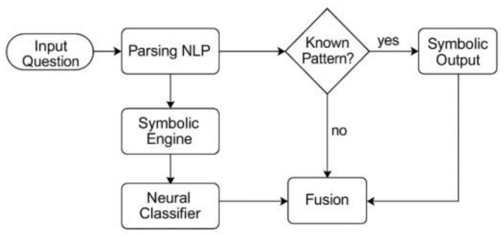
TF-IDF vectorization: to identify discriminative terms in problem formulations.

Semantic clustering: to group similar problems into decomposition structure classes.
Association rule mining: to detect stable co-occurrences between textual expressions and types of subproblems.
Dynamic rule base enrichment: to automatically inject newly learned rules into the decomposition engine, with optional manual or automated validation.
Application to the Bitcoin Example:
Let us return to the complex problem "How long can Bitcoin resist?", which was previously decomposed into ten subproblems (see Section 5.2). If several alternative formulations of this problem are manually annotated—e.g., What is Bitcoin’s future under strict regulations?, How will Bitcoin behave during a financial crisis?, Is Bitcoin technologically sustainable?—the system can identify recurrent structural correspondences:
The phrase "financial crisis" is frequently associated with the sub-task of macro-economic forecasting, leading to the activation of the AgentCrisisForecast.
Expressions like "evolve against competitors" often correlate with crypto comparison tasks, activating AgentAltcoins.
Structures involving "sustainability", "energy consumption", or "environmental impact" systematically trigger AgentEnergyFootprint.
Based on such correspondences, the system can derive generalized inductive rules, such as:
IF [concept = Bitcoin] AND [keyword = "crisis"] → likely subproblem = economic prediction → target agent = AgentCrisisForecast
IF [verb = "compare"] OR [term = "Ethereum", "Solana"] → task = inter-crypto analysis → agent = AgentAltcoins
IF [lexical field = environment, energy, pollution] → task = energy footprint estimation → agent = AgentEnergyFootprint
These rules, semi-automatically extracted from the annotated corpus, augment the manually defined rule base. They enable the system to learn new decomposition behaviors by observing trends in real-world user formulations.
In this way, the symbolic learning module ensures both autonomous evolution and progressive expansion of the system's decomposition capabilities, while maintaining strong interpretability of its internal reasoning.

5.4 Neuro-Symbolic Hybridization

One of the major strengths of the Decomposition Agent lies in its ability to combine symbolic approaches (based on explicit rules) with more flexible and adaptive machine learning techniques. This neuro-symbolic hybridization enriches the decomposition logic with empirical signals while preserving strong interpretability.
The system implements a hybrid architecture in which:
symbolic rules ensure structural and semantic consistency of the decomposition,
while neural modules provide plausibility scores for term-to-subtask associations, based on contextual vector representations.
These learning layers can thus suggest new or atypical associations, which are then validated or rejected by the symbolic engine.
Techniques used:

Contextual embedding models (e.g., BERT): to represent problem segments in a dense semantic space.
Relation classifiers: to estimate the likelihood that a segment corresponds to a specific subproblem.
Hybrid pattern matching: integration of neural scores into symbolic rules as dynamic weighting factors.
Threshold control: a configurable confidence threshold filters the suggestions made by the neural model.
Application to the Bitcoin example:
Let us revisit the complex problem "How far can Bitcoin resist?", which has previously been decomposed into ten subproblems (see Section 5.2), ranging from economic resilience to environmental impact, including technical limitations, competition from altcoins, and legal regulations.
Based on various user queries (e.g., Can Bitcoin survive a massive price crash?, Will it remain dominant against newer cryptocurrencies?, Is its energy model sustainable over time?), the system uses hybrid mechanisms to enhance its decomposition capabilities.
For example:
• The phrase "massive price crash" initially triggers a symbolic rule linked to a liquidity crisis. But the neural module refines the interpretation:
— Embedding is close to segments related to post-crash financial resilience, — High probability for the subtask crypto market crisis forecasting, → Joint suggestion to activate AgentCrisisForecast with a weighted confidence factor.
• The expression "remain dominant against newer cryptocurrencies" is covered by a symbolic rule linking competition to comparative analysis.
— The neural model confirms the semantic proximity with previous cases involving Ethereum, Solana, Cardano, → Reinforced activation of AgentAltcoins.
• The phrase "sustainable energy model" activates by default a symbolic rule based on the lexical field of energy and environment.
— The neural classifier reinforces the hypothesis by recognizing contextual elements such as Proof of Work, electrical consumption, carbon efficiency, → Robust confirmation of the activation of AgentEnergyFootprint.
These interactions allow the induction of adaptive hybrid rules, such as:
IF [concept = Bitcoin] AND [segment ≈ crash, collapse, massive drop]
→ subproblem = crypto financial resilience
→ agent = AgentCrisisForecast
IF [named entity = Ethereum OR altcoin] AND [verb ≈ compete, surpass]
→ subproblem = comparative crypto analysis
→ agent = AgentAltcoins
IF [lexical field = energy, pollution, sustainability]
→ subproblem = ecological footprint estimation
→ agent = AgentEnergyFootprint
Thus, neuro-symbolic hybridization provides:
enhanced robustness in the face of user formulation variability,
dynamic interpretation of ambiguous contexts,
and continuous evolution of the decomposition engine based on real-world inputs.
It embodies the core objective of S-AI: to bridge the explanatory rigor of symbolic reasoning with the contextual flexibility of neural inference, while remaining lightweight, interpretable, and computationally efficient.



UML diagram of the hybrid reasoning process

Figure 5.4 – UML diagram illustrating the neuro-symbolic fusion mechanism in the decomposition process. The diagram depicts the parallel processing of a query by the symbolic engine and the neural classifier, followed by a confidence-based comparison and the adaptive selection of the most relevant sub-task according to reliability scores.

5.5 Hormonal Integration in the Functioning of the Decomposition Agent (updated version with MetaAgent mediation)

As the cognitive entry point of the S-AI system, the Decomposition Agent plays a central role in the initial regulation of complex problem processing. In this capacity, it is fully integrated with the artificial hormonal signaling mechanism, through a dual functional coupling:

It can signal to the MetaAgent a strategic need based on the nature and perceived complexity of the detected problem.

It can also adapt its segmentation strategies in response to hormonal signals received from Gland Agents, via the system's signaling engine.

Hormone Emission Mechanisms (via the MetaAgent)

When the input problem is identified as critical, ambiguous, or highly transversal, the Decomposition Agent notifies the MetaAgent of the situation. Based on global system context, the MetaAgent may then decide to activate one or more Gland Agents. These agents may synthesize and emit: • A high-complexity hormone, indicating the need for fine-grained orchestration, • An alert hormone, informing the system that increased vigilance or resource reallocation is needed. This anticipatory signaling enables S-AI to proactively prepare by pre-activating relevant agents, reallocating computational resources, or adjusting internal trust thresholds.

Hormonal Reception and Strategic Adaptation

Conversely, the Decomposition Agent can receive hormonal signals indicating:
Temporal pressure (e.g., tempo_fast),
System overload (e.g., cpu_load_high),
Energy efficiency constraints (e.g., low_energy_mode).

Upon receiving such signals, it dynamically adapts its behavior: by prioritizing symbolic rule-based strategies, limiting the use of costly neural components, selecting faster or approximate decomposition patterns.

In this framework, hormonal signals are never emitted directly by the Decomposition Agent. Instead, they are exclusively synthesized by Gland Agents, upon explicit request by the MetaAgent (see Section 3.4). These agents prepare enriched hormonal profiles encoding operational constraints—such as urgency, precision trade-offs, or energy-saving modes—based on strategic intent. The Decomposition Agent interprets these hormones as contextual directives, adjusting its segmentation policy accordingly.

This hierarchical delegation ensures coherence and reinforces modularity by allowing global adaptation through low-coupling hormonal diffusion rather than direct reprogramming of cognitive components.
Bitcoin Example: Hormonal Modulation in Action
Let us revisit the problem:
“How long can Bitcoin resist?”
If this problem is received in a neutral context, the Decomposition Agent may apply: symbolic rules (Section 5.2), lightweight classifiers (Section 5.4), and, when appropriate, activate deep agents (e.g., AgentCrisisForecast, AgentScenarioSim).
However, in a hormonally constrained context:
Hormones received: low_ energy_mode + cpu_load_high → The agent restricts the decomposition to 6 simple subproblems (e.g., price history, volatility, sentiment), → skips the activation of deep learning agents (e.g., GANs, LSTM), → and favors low-cost symbolic segmentation.
In contrast, in a crisis or system-wide alert situation: • Need signaled: high_complexity_alert (from the Decomposition Agent) → The MetaAgent calls upon a Gland Agent, which emits the hormone system-wide,
→ The Decomposition Agent receives it and selects a strategy maximizing semantic coverage, even at the cost of longer processing time.
This hormonal modulation capability enables the Decomposition Agent to behave in a context-aware and self-regulating manner, while preserving the modular and interpretable architecture of the system. It reflects one of the core principles of Sparse AI: dynamically adapt computational effort in response to external constraints, without compromising reasoning quality or structural integrity.

5.6 Dynamic Knowledge Base for Adaptive Decomposition

The Decomposition Agent (DA) in the S-AI architecture relies on a dynamic knowledge base (KB-D) to perform task decompositions that are relevant, contextualized, and aligned with the capabilities of existing Specialized Agents. This knowledge base contains task schemas, input-output templates, heuristic rules, and segmentation patterns derived from past experiences or expert input.
Unlike the static, embedded knowledge bases of Specialized Agents, the KB-D is evolving, transversal, and centralized. It may include:
annotated task graphs,
functional ontologies to match subtasks with agent competencies, and performance metadata to optimize decomposition strategies based on feedback.
This knowledge base evolves through two complementary channels: via symbolic or neuro-symbolic mechanisms that automatically extract recurrent decomposition patterns from solved cases;
and through explicit contributions from human experts (e.g., physicians, engineers, architects), who can manually describe complex decompositions as graphs, symbolic rules, or domain-specific reasoning chains.
Each human contribution is stored with full traceability and can serve as a reusable model or reference for future decomposition suggestions. Users may accept, refine, or reject automatically generated proposals, establishing a symbolic human-AI collaboration loop.
The KB-D thus plays a central role in the intelligent plasticity of the Decomposition Agent. It enables the system to dynamically reconfigure its segmentation strategies while maintaining consistency, explainability, and computational efficiency—key pillars of the S-AI architecture.

5.7 Interaction with the MetaAgent (Developed Version)
The Decomposition Agent maintains a close and continuous interaction with the MetaAgent, forming a core component of the adaptive feedback loop essential to the robustness of the S-AI system. This bidirectional exchange is grounded in an intelligent dialogue, where each component enriches the other by sharing strategic information, contextual signals, and execution feedback.
On one hand, the MetaAgent can:
propose new segmentation rules based on the analysis of aggregated results, dynamically refine decomposition strategies, taking into account past performance or current constraints (such as time, energy, or precision),
signal recurring errors in classification or fragmentation, suggesting updates to the heuristics used by the Decomposition Agent.
On the other hand, the Decomposition Agent can: inform the MetaAgent of atypical, ambiguous, or poorly covered situations under current rules, emit strategic need signals when it detects cognitive overload, semantic dead-ends, or insufficient granularity,
provide semantic summaries of its decompositions, enabling the MetaAgent to optimize the selection of Specialized Agents or to adapt the system's hormonal configuration.
This synergy enhances the functional co-evolution between the two agents: with each iteration, the MetaAgent improves its global orchestration capabilities, while the Decomposition Agent enhances the relevance, efficiency, and precision of its segmentations. Together, they contribute to the system’s self-adaptation when faced with complex and evolving problems.
Furthermore, any hormonal coordination involving the Decomposition Agent is mediated through the MetaAgent. When activation of a specific hormonal profile is required (e.g., high_complexity, low_energy_mode), the Decomposition Agent issues a request to the MetaAgent, which then decides whether to trigger a Gland Agent if necessary (see Sections 5.5 and 3.4). This delegation chain ensures coherence in strategic intent across the system.
Ultimately, this interaction represents more than a simple data exchange: it embodies a form of distributed collective intelligence, grounded in self-assessment, adaptive memory, and contextual orchestration. It illustrates the cooperative and hierarchical functioning of the S-AI system.

5.8 Demonstration Code
A modular prototype of the Decomposition Agent has been implemented using standard Python tools. The code integrates symbolic rules, TF-IDF classification, and neuro-symbolic fusion mechanisms. It provides a minimal working example of adaptive task decomposition. The detailed pseudocode of this agent is provided in Section 11, which is dedicated to the technical implementation of the S-AI system.

5.9 Discussion
The Decomposition Agent is more than a technical component. It encapsulates the philosophy of S-AI: divide to simplify, activate to minimize, and learn to evolve. By structuring intelligence around dynamic decomposition, it enables lightweight AI to operate with greater precision, transparency, and sustainability.
The decomposition techniques presented in this section build upon methodological components that, taken individually, already have counterparts in existing literature. The symbolic rule-based decomposition (section 5.2) extends classical parsing and conditional routing approaches widely used in expert systems

and task-oriented architectures. Rule learning from annotated corpora (section 5.3) aligns with supervised symbolic induction methods commonly employed in explainable AI systems. Lastly, the neuro-symbolic hybridization strategy (section 5.4) draws inspiration from recent efforts to combine the precision of symbolic reasoning with the robustness of lightweight neural models. However, the originality of the S-AI approach does not lie in the intrinsic novelty of each mechanism, but rather in their synergistic integration within a modular agent-based architecture governed by a strict principle of activation parsimony. The decomposition process is no longer treated as a static preprocessing module, but as a dynamic cognitive component, capable of adapting its strategy to the context (through artificial hormonal signaling), evolving over time (via continuous rule enrichment), and coordinating with the MetaAgent to optimize the activation of specialized agents. This modular, explainable, and resource-efficient orchestration enables the S-AI system to address complex tasks with minimal computational overhead, while preserving high levels of logical traceability in the decision-making process. This positioning fundamentally distinguishes S-AI from traditional approaches, placing it at the intersection of symbolic engineering, adaptive learning, and context-aware orchestration.

Section 6 – The MetaAgent

6.1 – Introduction of the MetaAgent

The MetaAgent occupies a central position within the S-AI architecture. It serves as the core decision-making and orchestration unit of the system. Its primary role is to supervise the selection, activation, and coordination of specialized agents, based on task context, complexity, and historical performance. Through this dynamic management process, the MetaAgent ensures that the S-AI system remains parsimonious, adaptive, and continuously evolving, improving over time via feedback loops.

6.2 – Functional Role of the MetaAgent (adapted version with Gland Agents)

The MetaAgent fulfills several key functions within the S-AI system:
Analyzing subtasks received from the Decomposition Agent;
Selecting the most suitable specialized agents based on task profiles;
Activating or deactivating agents dynamically, depending on contextual constraints;
Supervising agent performance during task execution;
Guiding result aggregation strategies;
Performing adaptive learning based on feedback from previous operations.
In addition to these roles, the MetaAgent is also in charge of interacting with Gland Agents. When specific contextual conditions require internal modulation—such as increased complexity, energy conservation, or urgency—the MetaAgent triggers the activation of one or more Gland Agents. These agents release artificial hormonal signals that influence the behavior of other components in the system, allowing global coordination and adaptive regulation across specialized agents and decomposition mechanisms.
By centralizing these functions, the MetaAgent acts as the strategic brain of the S-AI system, enabling intelligent orchestration and efficient resource management.

6.3 – Artificial Hormonal Signaling and Propagation

6.3.1 Introduction

As part of the Sparse Artificial Intelligence (S-AI) architecture, we introduce an artificial hormonal signaling mechanism inspired by the biological functioning of living organisms. This mechanism endows the system with contextual adaptive capabilities, enabling it to dynamically respond to variations in its global state (e.g., urgency, computational load, confidence). This module empowers the MetaAgent, as the system's central orchestrator, to fine-tune coordination strategies, while allowing specialized agents to modulate their operational behavior accordingly. Concretely, this signaling manifests through the emission of artificial hormones—such as urgency, high_load, or confidence_bias—that influence the intensity or processing mode of each subproblem.

6.3.2 Signal Propagation Mechanism

The system integrates two main sources of hormonal signals:
A global hormonal context, simulated dynamically at each evaluation cycle;
A persistent signaling engine (HormonalEngine) that injects and propagates explicit signals based on observed conditions.
These signals are accessed by the MetaAgent, which uses them to assess task complexity and may trigger fast_mode activation among agents sensitive to urgency or computational constraints.
In addition to these sources, a third functional entity plays an increasingly important role in hormonal signal synthesis: the Gland Agents (see Section 3.4). These agents serve as intermediaries that translate high-level orchestration intents—emitted by the MetaAgent—into concrete hormonal messages. Each gland is associated with a predefined behavioral profile and can embed contextual parameters such as urgency thresholds, energy sensitivity, or task prioritization levels directly into the hormones it produces. These enriched signals are then injected into the HormonalEngine, where they propagate throughout the system like any other hormone. This modularization of signal synthesis enables the MetaAgent to delegate low-level contextual modulation, reinforcing architectural parsimony and supporting scalable adaptation.

6.3.3 Illustration of the Hormonal Process

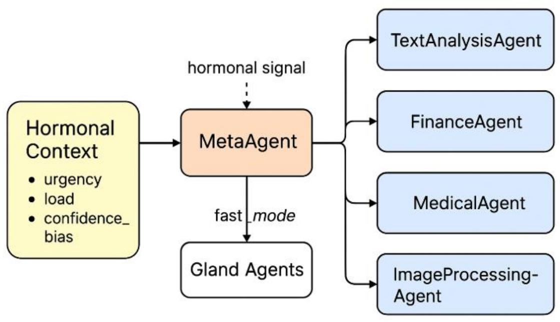


Figure 6.3.3 – Artificial Hormonal Signaling in the S-AI System

6.3.4 Discussion

Hormonal signaling represents a major advancement in system autonomy and resilience. It enables dynamic resource regulation, reducing reliance on heavy models when conditions demand efficiency. This mechanism strengthens the synergy between bio-inspiration and computational frugality—two core principles of the S-AI paradigm.
Preliminary experiments indicate that hormonal modulation improves system responsiveness without significant performance degradation in simple or moderately complex cases. It offers a powerful lever for building a sparse, reactive, and robust artificial intelligence.

6.4 – Agent Selection Mechanisms

The MetaAgent is responsible for selecting which specialized agents to activate based on the nature and complexity of each subtask. This selection process directly depends on the decomposition structure generated upstream by the Decomposition Agent (see Section 5). It is based on this segmentation that the MetaAgent identifies the elementary subtasks, their typology, and the relevant agent classes best suited to address them efficiently.
This selection is guided by semantic task analysis, a competence matrix of available agents, historical performance data, and expected cost-benefit trade-offs.
A dynamic scoring mechanism ranks agents according to their relevance, estimated efficiency, and resource consumption. Only the most suitable agents are activated, in line with the principle of activation parsimony. These decisions rely on adaptive heuristics and reinforcement learning-inspired mechanisms, allowing the MetaAgent to progressively refine its strategy.
This selection process is also influenced by contextual signals transmitted through the artificial hormonal system. When signals related to urgency, system overload, or heightened alert are detected, the MetaAgent dynamically adjusts its selection criteria—for example, by prioritizing faster or low-cost agents, or conversely, by activating expert agents in critical situations. This coupling with the hormonal engine ensures context-aware orchestration aligned with the global system state.
In some cases, this modulation does not originate directly from the MetaAgent but is prepared upstream by a gland agent (see Section 3.4). The gland encodes a targeted behavioral profile that implicitly defines preferred agent families, execution constraints, or adaptive priorities. Informed by this enriched signal, the MetaAgent can fine-tune its agent selection strategy without managing all low-level contextual parameters directly, thereby enhancing the scalability and modularity of decision-making.
By combining these factors with the adaptive memory described in Section 6.5, the MetaAgent develops a strategic reflexivity that considers past performance, present context, and ongoing hormonal modulation. This triple integration—semantic, adaptive, and hormonal—endows the agent selection mechanism with cognitive robustness, essential for operation in open, constrained, or dynamic environments.

6.5 – Adaptive Learning through Feedback Loops

A core functionality of the MetaAgent is its ability to learn and adapt progressively through continuous feedback loops. These mechanisms allow the system to dynamically evolve and refine its behavior over time, ensuring optimal task resolution and resource usage.
During each subtask execution, the MetaAgent collects a set of key performance indicators, including:
The execution time of specialized agents;
The quality of the output produced;
The confidence level associated with the activated agent.
This information is stored in a dynamic performance memory, serving as the basis for adaptive learning. The MetaAgent continuously recalibrates its agent selection strategies, reweights agent priorities, and updates activation rules based on contextual efficiency and historical results.
Additional indicators such as task success rates, computational cost, and impact on final aggregation also contribute to the evaluation process. Underperforming agents can be deprioritized, updated, or replaced, ensuring the system maintains high levels of efficiency, scalability, and parsimony.

Through this ongoing learning process, the S-AI system becomes smarter and more efficient with each execution cycle, reinforcing its ability to operate intelligently while minimizing unnecessary activation of resources.

Furthermore, performance feedback not only enhances agent selection but also contributes to the co-evolution of decomposition strategies managed by the Decomposition Agent. As the MetaAgent continuously learns from execution results, it can influence how complex tasks are segmented, suggesting refinements in granularity or subtask boundaries. This ensures that decomposition remains aligned with real-time agent capabilities and system performance, creating a dynamic synergy between analysis and orchestration.

Beyond simple adjustment of decisions, the MetaAgent develops a form of long-term strategic learning based on the systematic exploitation of past experiences. For each problem-solving instance, the actions taken (agents selected, orchestration order, applied weightings) are correlated with the obtained outcomes. These associations are stored as contextual performance profiles, enabling the MetaAgent to gradually refine its agent selection mechanisms.

This learning process does not rely on deep learning, but rather on a weighted adaptive memory, in which past strategies are reinforced, weakened, or neutralized depending on their contextual effectiveness. This type of learning is explainable, traceable, and fully aligned with the principles of computational parsimony. It transforms the MetaAgent into a self-evolving orchestrator, capable of optimizing its decision-making over time without sacrificing modularity or interpretability.

6.6 – Global Orchestration Capabilities of the MetaAgent

Beyond its role in selecting and supervising specialized agents, the MetaAgent is also responsible for the global orchestration of the entire S-AI system, including the governance of the Decomposition Agent, specialized expert agents, result aggregation modules, and the Result Access Module (RAM). It acts as the cognitive conductor of the architecture, coordinating both the analysis process (decomposition) and the execution process (task resolution), while ensuring systemic coherence and activation parsimony.

A functional and hierarchical duo: MetaAgent & Decomposition Agent

The Decomposition Agent operates as the upstream interface of the system. It is responsible for analyzing complex problems and segmenting them into elementary subtasks using symbolic rules, domain-specific heuristics, and hybrid neuro-symbolic techniques. This decomposition is not a mere mechanical split but a structured cognitive process, accounting for semantics, dependencies, and processing constraints.

However, this operation occurs only under the explicit supervision of the MetaAgent. The MetaAgent:

Receives the initial problem input;

Decides whether decomposition is necessary;

Defines the contextual parameters (granularity level, segmentation constraints);

Activates the Decomposition Agent accordingly.

Once the decomposition is complete, the MetaAgent resumes control, evaluating the relevance and structure of the subtasks, validating or rejecting certain elements, requesting refinements if needed, and most importantly, orchestrating the allocation of each subtask to the most appropriate specialized agent.

This articulation guarantees:

A meaningful segmentation of complex problems;

An optimized activation of system resources;

And a seamless coherence between analysis (decomposition) and action (orchestration).

The MetaAgent as an adaptive decision-making entity
Based on the validated subtasks, the MetaAgent assesses their complexity, priority, interdependencies, and contextual constraints. It selects the most suitable specialized agents or invokes expert agents for high-complexity tasks. It supervises execution time, inter-agent communication flows, the quality of intermediate outputs, and feedback loops that refine its own selection strategies.
The MetaAgent does not execute tasks directly but acts as a distributed executive brain that observes, analyzes, learns, adapts, and optimizes.
An evolving and parsimonious system regulator
The MetaAgent also functions as an adaptive regulator of the S-AI system. It relies on:
A memory of past agent performance;
A multi-criteria optimization grid (cost, reliability, efficiency);
Context-sensitive activation rules that evolve over time;
Self-adjustment mechanisms based on aggregation outcomes and user feedback from the RAM.
In this orchestration role, the MetaAgent also delegates low-level modulation to Gland Agents. When strategic conditions such as urgency, low energy availability, or increased complexity arise, the MetaAgent triggers the corresponding Gland Agent. These agents synthesize and inject contextual hormonal signals into the system, which modulate the behavior of decomposition and specialized agents accordingly. This delegation mechanism reinforces the modularity and scalability of orchestration, while preserving the MetaAgent's focus on high-level strategic decisions.
This adaptive orchestration ensures intelligent, fluid, and parsimonious resource management, in alignment with the S-AI paradigm. The MetaAgent thus embodies the systemic and evolving intelligence of Sparse AI: a system designed to minimize computational load while maximizing operational relevance, in accordance with the principle:“Activate less, but activate better.”

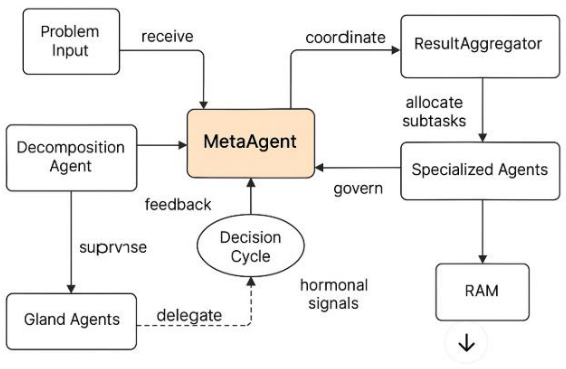


Figure 6.6 – UML Diagram: Global Orchestration by the MetaAgent

6.7 – Hormonal Regulation by the MetaAgent

As the orchestrator of the S-AI system, the MetaAgent plays a central role in artificial hormonal regulation—not as a direct emitter of hormones, but as the strategic coordinator of their deployment. It acts as a regulator, modulator, and interpreter of the hormonal signals circulating within the architecture. This capability enables it to orchestrate global contextual modulations that influence the behavior of other system entities (specialized agents, decomposition agent, etc.) via the Gland Agents.

For instance, when the MetaAgent detects a critical time constraint or computational overload, it triggers the appropriate Gland Agent, which synthesizes a hormonal signal such as urgency or energy_saving. This enriched signal is then injected into the system through the HormonalEngine, where it propagates to relevant components. These hormones can lead to the activation of alternative behaviors: faster processing modes, reduced task granularity, or bypassing of resource-intensive models.
Conversely, the MetaAgent also integrates the ambient hormonal climate into its orchestration strategy. A high concentration of certain hormones may influence agent selection, task prioritization, or the postponement of low-priority operations. It thus becomes capable of resonating with the internal state of the system, much like an intelligent homeostat operating at the cognitive level.
This hormonal layer significantly enhances the plasticity of the MetaAgent. It supplements explicit decision-making with soft, context-aware modulation, which is essential for adapting to dynamic, multi-domain environments with fluctuating constraints.
In most cases, the MetaAgent does not directly emit hormonal signals, but delegates this function to dedicated Gland Agents (see Section 3.4). When contextual modulation is required—such as activating a low_energy mode or responding to a critical event—the MetaAgent formulates a strategic intent, which it transmits to the corresponding Gland Agent. The latter encodes this intent into an enriched hormonal signal, embedding parameters such as duration, intensity, and the targeted agent group. The hormone is then injected into the system and propagated using standard HormonalEngine mechanisms.
This clear separation of responsibilities allows the MetaAgent to focus on high-level orchestration, while relying on specialized components for fine-grained, contextual modulation. It reinforces the modularity, scalability, and parsimony of the S-AI architecture, fully aligned with its foundational principles.

6.8 Strategic Knowledge Base and Meta-Reflectivity of the MetaAgent

The MetaAgent in the S-AI system acts as the central orchestrator, and its effectiveness relies heavily on its ability to leverage a strategic knowledge base (KB-MA). This knowledge base goes beyond static orchestration rules: it serves as a dynamic repository of experiences, contextual factors, historical performance metrics, and system-level preferences, which fuels its adaptive decision-making capabilities.
The KB-MA includes:
profiles of Specialized Agents: domains of expertise, efficiency records, and optimal activation conditions;
empirically validated selection strategies, adjusted according to task types and system configuration;
models of agent-context interaction enabling the anticipation of synergies, conflicts, or redundancies between agents.
This cognitive substrate provides the MetaAgent with a form of operational meta-reflectivity: it does not merely follow predefined rules, but dynamically adapts its decisions based on accumulated strategic memory.
The enrichment of this knowledge base is primarily achieved through:
feedback-driven learning loops,
observation of aggregated outcomes in similar contexts,
and reinforcement of successful strategies through historical comparison.
The KB-MA thus transforms the MetaAgent into an intelligent and evolving control node, capable of coordinating a complex multi-agent system while upholding the core principles of S-AI: parsimony, adaptability, and explainability.
In this evolving architecture, gland agents also contribute to the MetaAgent’s strategic reasoning. Each gland maintains a lightweight memory of previously synthesized hormonal profiles and their contextual

effectiveness. This information can be referenced by the MetaAgent to select not only the appropriate agent set, but also the most relevant modulation strategy for a given situation. In doing so, the MetaAgent leverages the distributed memory of glands to refine not just execution patterns, but also the preparatory hormonal context, reinforcing its meta-reflective capabilities while delegating low-level encoding.

6.9 – Implementation and Illustrative Code

The implementation of the MetaAgent requires an efficient orchestration framework capable of managing dynamic agent selection, performance tracking, and adaptive learning. Below is a simplified Python prototype to illustrate key concepts:

```
# MetaAgent prototype – adaptive performance-based selection
class MetaAgent:
    def __init__(self, agent_profiles):
        self.agent_profiles = agent_profiles
        self.performance_log = {}

    def select_agents(self, subproblem):
        candidates = [a for a in self.agent_profiles if subproblem["type"] in self.agent_profiles[a]]
        scored = {a: self.evaluate(a, subproblem) for a in candidates}
        return sorted(scored, key=scored.get, reverse=True)

    def evaluate(self, agent, subproblem):
  past = self.performance_log.get(agent, {"score": 0.7})
        return past["score"]

    def feedback(self, agent, result_quality):
 self.performance_log.setdefault(agent, {"score": 0.7})
        delta = 0.1 if result_quality == "good" else -0.1
        self.performance_log[agent]["score"] = min(max(self.performance_log[agent]["score"] + delta, 0.0),
1.0)
```

This basic prototype demonstrates the idea of adaptive scoring and feedback integration. A full implementation would include advanced strategies, hybrid learning models, and tight coupling with the Decomposition Agent.

6.10 – Discussion

The MetaAgent as Cognitive Regulator

The MetaAgent embodies the philosophy of intelligent, context-aware orchestration at the core of Sparse Artificial Intelligence. It ensures selective agent activation, experience-based learning, and modular governance over distributed decision processes. Far from being a static planner, it acts as a self-regulating cognitive layer, continuously adapting orchestration strategies based on semantic input, agent feedback, and contextual signals, guiding the system toward sustainability, scalability, and explainability.

Its orchestration capabilities are further enhanced by the integration of artificial hormonal signaling, which introduces an additional layer of soft coordination and low-cost behavioral modulation. While the

MetaAgent does not emit hormones directly, it strategically delegates this function to Gland Agents, which act as biochemical translators of intent. These agents encode high-level orchestration commands into enriched hormonal messages that propagate through the system, influencing specialized agents and decomposition mechanisms without requiring explicit task reassignment. This delegation reinforces the architectural modularity and allows the MetaAgent to focus on strategic decision-making.
By combining semantic task decomposition, agent profiling, performance tracking, and context-sensitive learning, the MetaAgent delivers a robust, intelligent, and evolvable orchestration strategy that defines the operational core of the S-AI model.
Looking ahead, several avenues can further extend the MetaAgent's capabilities:
Integration of multi-agent negotiation protocols, enabling the MetaAgent to arbitrate between conflicting agent preferences;
Use of meta-reasoning loops, allowing it to reflect on its own orchestration strategies and refine them autonomously;
Development of a dynamic gland agent allocation mechanism, where gland roles evolve based on system needs or agent ecology;
Implementation of cross-domain orchestration heuristics, leveraging accumulated knowledge across tasks to better anticipate future activations.
In sum, the MetaAgent is not merely the executive module of the S-AI system—it is its strategic conscience, capable of evolving with the system it governs. Its design reflects the fundamental ambition of S-AI: to activate less, but activate wisely.

Section – 7 Gland Agents – Contextual Modulators Inspired by the Endocrine System

7.1 Bio-Inspired Justification and Regulatory Role

As a natural extension of the artificial hormonal signaling mechanism introduced in the architecture of S-AI (see Section 2.5), gland agents constitute a core component of the system, designed from the outset to fulfill a role of contextual and adaptive regulation.
Inspired by biological endocrine glands, these entities act as preparatory and secretory modules for artificial hormones. Each gland is associated with a specific behavioral profile (exploration, resilience, critical reasoning, etc.) and is triggered by the MetaAgent when a global reconfiguration of the system's behavior is required.
This mechanism introduces a functional dissociation:
the strategic decision remains the responsibility of the MetaAgent;
the operational preparation of execution parameters is handled by the glands.
This separation supports the distributed modularity principle of S-AI, reducing the central load of orchestration while enabling scalable context-aware adaptation.

7.2 Systemic Operation of Gland Agents

Gland agents are activated when the system needs to adopt a new global behavioral profile. Their internal functioning follows these key stages:
Command from the MetaAgent, expressing an intent to activate a particular behavior (e.g., "activate EnergySaving profile").
Gland processing, during which the appropriate elements are prepared:
contextual data required for the profile;

modulation parameters (thresholds, priorities, attentional filters);
the type, intensity, and duration of the artificial hormone to be secreted.
Hormone secretion via the HormonalEngine, which diffuses the enriched hormone across the system.
Activation of specialized agents, who adjust their behavior dynamically based on the hormone content they receive.
These hormones can act in stimulatory, inhibitory, or selective ways, enabling seamless behavioral reconfiguration of the system without requiring the MetaAgent to micromanage all adaptations.

7.3 Illustrative Use Cases of Hormonal Profiles Activated by Glands

The gland-hormone mechanism proves particularly effective in situations requiring global behavioral reconfiguration. Several real-world inspired use cases illustrate its relevance:

a) Multi-situational Autonomous Robot

An autonomous robot operating in an unknown environment may switch between exploration, retreat, stationary analysis, or communication. Each phase requires activating different sensors, movement parameters, and energy limits.
The "Exploration" gland configures navigation thresholds, selects active sensors, and secretes a hormone that signals relevant agents to adopt the appropriate configuration.

b) Multi-style Conversational AI

Conversational systems may alternate between different response tones: pedagogical, humorous, formal, or empathetic.
The "PedagogicalStyle" gland injects lexical filters, syntactic simplifications, and knowledge adaptation levels into the hormone. Language agents interpret this hormone and adjust their outputs accordingly.

c) Contextual Multi-Criteria Decision System

In economic crisis scenarios, the system must adopt a resilient decision profile: slower prediction cycles, increased uncertainty tolerance, temporary deactivation of certain speculative agents.
The "EconomicResilience" gland reduces decision confidence thresholds, suppresses speculative behaviors, and modifies sensitivity to economic signals—all encapsulated within a single hormone emission.

d) Multi-phase Autonomous Drone Mission

A drone alternates between fast travel, fixed environmental observation, encrypted transmission, and stealth evasion.
Each phase corresponds to a specific gland, which delivers the appropriate operational parameters through hormones to agents controlling propulsion, sensors, and communication.

e) Multi-agent Simulation with Adaptive Strategy

In simulations of social or biological systems, agents need to adapt strategies depending on stress, resource availability, or social tension.
A "SocialStress" or "Abundance" gland emits hormones that modulate collective behavior—such as cooperation level, risk tolerance, or aggressiveness—across all participating agents.

7.4 Role within the S-AI Architecture

Gland agents are situated between the MetaAgent and the HormonalEngine, forming an intermediate physiological layer within the S-AI architecture (see functional diagram in Section 3.6). Their contribution enables:

Delegation of profile preparation away from the MetaAgent;
Physiological modularity, where each gland encapsulates a reusable operational configuration;
Contextual collective responsiveness via diffuse hormonal signaling;
Simplified system-wide reconfiguration, with a single intention triggering system-wide adaptation.
Glands thus reinforce the S-AI principles of biological inspiration, distributed control, and activation parsimony, enabling graceful adaptation to diverse situations with minimal computational overhead and maximum interpretability.

7.5 Illustrative Example: AdrenalineGlandAgent in Autonomous Combat Aircraft Systems

To demonstrate the functional relevance of gland agents in high-stakes adaptive systems, we introduce the AdrenalineGlandAgent, a digital endocrine module designed for intelligent combat aircrafts. In this context, gland agents modulate the internal state of the AI system in response to dynamic battlefield signals, such as incoming threats, system fatigue, or cognitive overload.
The AdrenalineGlandAgent monitors key contextual variables (e.g., threat_level, cognitive_fatigue) and injects artificial "adrenaline" into the system when critical thresholds are crossed. This hormone modulates the behavior of specialized agents, particularly those involved in decision-making under pressure, such as a CombatTacticianAgent.
For instance, when the threat_level exceeds 0.7, the gland secretes a high dose of adrenaline. The CombatTacticianAgent, sensitive to this hormone, switches to a highly aggressive tactical mode, prioritizing rapid and risky maneuvers. This architecture enables the system to react adaptively in real-time, much like a biological organism facing extreme conditions.
Such a mechanism simulates a form of context-aware emotional reflex, bridging symbolic reasoning with low-latency behavioral shifts. The hormonal logic complements the modular decomposition by adding a transversal layer of homeostatic regulation, allowing agents to dynamically align their actions with the global situational pressure.
This use case highlights the potential of artificial gland agents to inject adaptive tension and modulate cognitive urgency across distributed systems, thus enhancing robustness, responsiveness, and realism in autonomous decision-making.

7.5.1 Code Illustration: Context-Aware Hormonal Modulation

To materialize the architecture described above, we provide a concrete implementation of the AdrenalineGlandAgent. This gland agent monitors the contextual variables of the system and secretes artificial "adrenaline" through the hormonal engine when predefined thresholds are crossed.

```
class AdrenalineGlandAgent:
    def __init__(self, hormonal_engine):
        self.hormonal_engine = hormonal_engine
        self.threshold = 0.7

    def evaluate_and_secrete(self, context):
        threat = context.get("threat_level", 0.0)
        fatigue = context.get("cognitive_fatigue", 0.0)

        if threat >= self.threshold:
            self.hormonal_engine.secrete("adrenaline", intensity=threat)
        elif fatigue >= 0.8:
            self.hormonal_engine.secrete("adrenaline", intensity=fatigue * 0.5)
```

This gland is instantiated in the system's main loop and invoked during execution. In parallel, a specialized agent such as the CombatTacticianAgent adjusts its behavior depending on the current adrenaline level:

```
class CombatTacticianAgent:
    def __init__(self, hormonal_engine):
        self.hormonal_engine = hormonal_engine

    def act(self, context):
        adrenaline = self.hormonal_engine.get_level("adrenaline")

        if adrenaline >= 0.7:
            self._aggressive_attack()
        elif adrenaline >= 0.3:
            self._standard_engagement()
        else:
            self._defensive_maneuver()
```

The coupling between the gland and the agent enables low-latency reflex adaptation, orchestrated by the internal hormone dynamics. This example showcases the ability of S-AI to incorporate real-time endocrine-style modulation across its decision layers.

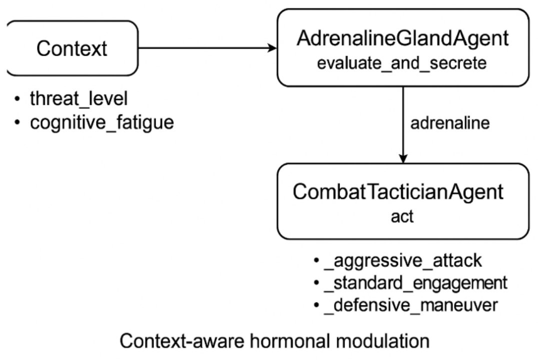


Figure 7.5 – Contextual Hormonal Modulation by AdrenalineGlandAgent

This diagram illustrates the flow of hormonal modulation within an autonomous combat aircraft system. The contextual engine provides real-time variables such as threat_level and cognitive_fatigue, which are evaluated by the AdrenalineGlandAgent. When critical thresholds are reached, the gland secretes artificial adrenaline into the system. The CombatTacticianAgent then adjusts its tactical behavior based on the current hormone level, enabling fast switching between defensive, standard, and highly aggressive combat modes.

This architecture demonstrates how endocrine-style signaling allows for low-latency behavioral adaptation, integrating seamlessly with the modular and parsimonious principles of the S-AI system.

Section 8 – The Result Aggregator

81 Introduction

In a modular architecture such as Sparse Artificial Intelligence (S-AI), once the subproblems are processed by the Specialized Agents, their outputs must be synthesized into a coherent and semantically structured global answer. This critical task is performed by the Result Aggregator, which serves as the final layer of orchestration and ensures system-wide unity.

8.2 Functional Role of the Aggregator

The Result Aggregator is responsible for:
Collecting the outputs from the activated agents,
Synthesizing partial results,
Handling the heterogeneity of output formats (text, numbers, classifications),

Applying weightings based on agent reliability,
Ensuring overall coherence and resolving possible contradictions.
It is not a simple concatenation mechanism but rather an intelligent module for structured recomposition.

8.3 Types of Aggregation

Depending on the nature of the task, different aggregation modes can be used:

Aggregation Type	Description	Example
Textual Fusion	Summaries or concatenation of generated text	Multi-agent synthesis of paragraphs
Weighted Average	Fusion of numerical results with weight factors	Computing weighted score averages
Majority Vote	Classification based on agent consensus	Label choice by majority
Sequential Composition	Ordered chaining of sub-responses	Multi-step logical reasoning answers

8.4 Hormonal Modulation of Result Aggregation

The Result Aggregator can also benefit from artificial hormonal signaling to dynamically adjust its integration strategies. Depending on the system's global context, it can adapt how it merges the partial outputs produced by the active agents, in order to ensure more relevant, faster, or more robust final outputs. For example, the presence of an urgency hormone may prompt the Aggregator to apply a fast-track strategy, prioritizing immediately actionable responses. Conversely, a complexity hormone may activate a more refined integration mode, incorporating additional cross-weighting or consensus checks among agents.
The Aggregator may also include hormonal modulators in its scoring or acceptance functions, raising or lowering certain thresholds based on perceived contextual pressure. This ongoing adaptation allows for a plastic aggregation process, aligned with the system’s overall state without compromising the modular logic.

This approach enhances the S-AI system's ability to generate context-sensitive and adaptive outputs, while preserving core values such as traceability, interpretability, and computational efficiency.
In certain contexts, these hormonal configurations, although supervised by the MetaAgent, are exclusively prepared and released by gland agents (see Section 3.4). By encoding specific expectations related to the output level — such as processing speed, consensus robustness, or degree of abstraction — gland agents indirectly influence the behavior of the Aggregator. This intermediate signaling enables the Aggregator to align its strategy with high-level systemic intentions without requiring explicit reprogramming, thereby enhancing both the flexibility and parsimony of the result consolidation process.

8.5 Agent Weighting and Confidence Levels

To improve response quality, the Aggregator may incorporate agent confidence levels, as well as historical reliability indicators provided by the MetaAgent. This makes it possible to prioritize high-performing agents and reduce the influence of less reliable ones.
Illustrative formula:
final_score = (score_A * weight_A + score_B * weight_B) / (weight_A + weight_B)

8.6 Simplified Implementation Example

```
class ResultAggregator:
  def __init__(self):
    self.results = []
  def add_result(self, result, weight=1.0):
    self.results.append((result, weight))
  def aggregate(self, mode="weighted_average"):
    if mode == "weighted_average":
      total, total_weight = 0.0, 0.0
      for val, w in self.results:
        total += val * w
        total_weight += w
 return total / total_weight if total_weight > 0 else None
    elif mode == "majority_vote":
      from collections import Counter
      values = [val for val, _ in self.results]
      return Counter(values).most_common(1)[0][0]
    elif mode == "text_concat":
      return " ".join([val for val, _ in self.results if isinstance(val, str)])
```

8.7 MetaAgent-Guided Aggregation

The Aggregator can be guided by the MetaAgent, which may:
Propose aggregation strategies based on the context,
Dynamically adjust weightings,
Detect inconsistencies and request recomputation,
Integrate outputs into the system's learning loop.
This interaction enhances the adaptive intelligence and explainability of the system.

8.8 Discussion

The Result Aggregator completes the modular intelligence loop of the S-AI system by ensuring coherent synthesis of distributed reasoning. It illustrates that intelligence is not limited to individual agents, but also emerges through their dynamic coordination and integration. Through its selective weighted fusion capabilities, the Aggregator guarantees both efficiency and reliability in the final output of the system.

Section 9 – The Result Accessing Module (RAM)

9.1 Introduction

The Result Accessing Module (RAM) serves as the final interface between the S-AI system and its users. While the Result Aggregator merges individual outputs into a coherent response, the RAM is responsible for structuring, formatting, and delivering these results in a comprehensible and actionable form. This module plays a key role in user experience, transparency, and system traceability.
Beyond simple output delivery, the RAM can also provide contextual explanations, confidence scores, and traceability metadata, making the decision-making process more interpretable and trustworthy.

9.2 Structuring and Formatting Results

To ensure consistent output, the RAM applies a standardized structure adapted to various use cases:
Numerical outputs: Presented with confidence intervals where applicable.
Textual responses: Organized into structured summaries or bullet-point lists.
Hierarchical results: Grouped and ordered according to logical dependencies.
Multimodal outputs: Seamless integration (e.g., text + image + numerical data).
This adaptable formatting ensures that the results are intuitive and directly usable, regardless of the original query's complexity.

9.3 Confidence Evaluation and Explainability

A key component of the RAM is its explainability module, which provides:
Confidence levels for each sub-result.
A breakdown of agent contributions, indicating which agent handled which task.
Reasoning traces highlighting the steps that led to the final response.
This approach promotes transparency and trust, which are critical in sensitive domains such as healthcare, finance, or law.

9.4 Traceability and Reproducibility

To enhance transparency, the RAM maintains a log of past executions, including:
Initial queries,
Activated agents,
Intermediate results,
Final outputs,
Performance metrics.
This enables users to audit past results, verify reproducibility, and adjust their queries based on historical behavior.

9.5 Output Handling Implementation Example

```
class ResultAccessingModule:
    def __init__(self):
        self.log = []
    def process_result(self, result, confidence, explanation):
        formatted_output = {
            "Result": result,
            "Confidence": confidence,
            "Explanation": explanation
        }
        self.log.append(formatted_output)
        return formatted_output

    def get_history(self):
        return self.log
```

This implementation ensures that results are consistently formatted, logged, and accessible for future reference.

9.6 Hormonal Adaptation of Result Delivery

The RAM can integrate signals from the artificial hormonal signaling layer to dynamically adapt the format, level of detail, or presentation mode of final results. This functionality allows the RAM to modulate output delivery according to the global context of the system or the specific constraints of the user.

For example:

- An urgency hormone may trigger concise, high-level output by omitting explanatory details deemed secondary.
- A complexity hormone may prompt full display of reasoning traces, involved agents, and detailed confidence levels.
- A high system load hormone may temporarily disable heavy visual elements (infographics, images) in favor of optimized text-based rendering.

This hormonal regulation of the interface enhances the RAM’s contextual intelligence and flexibility. It ensures a dynamic match between output modality and real-world context, without requiring manual intervention. It directly contributes to the overall effort of S-AI to provide a parsimonious, context-aware, and user-centric AI.

In some cases, the hormones influencing output formatting or detail level do not originate directly from the MetaAgent but are synthesized by gland agents (see Section 3.4). These agents translate strategic output intentions—such as clarity, urgency, or enhanced explainability—into enriched hormonal signals diffused across the system. The RAM, sensitive to these signals, dynamically adjusts its presentation mode without requiring hard-coded rules, thereby reinforcing the modularity and autonomous adaptability of the output layer within S-AI

9.7 Discussion

The Result Accessing Module finalizes the S-AI pipeline by making results interpretable, structured, and traceable. By integrating confidence assessment, explainability mechanisms, and reproducibility tracking, it bridges the gap between modular artificial intelligence and real-world usability. This module ensures that S-AI not only optimizes computational efficiency, but also enhances human understanding and trust in its decision-making processes

Section 10– Experimental Evaluation

10.1 Test Scenarios and Datasets

To validate the efficiency and relevance of the Sparse Artificial Intelligence (S-AI) architecture, a series of experimental scenarios were designed to simulate realistic tasks of varying complexity. These scenarios span multiple domains, including numerical computation, text summarization, question answering (QA), and multi-agent reasoning.

The test dataset includes:

A set of standardized synthetic problems, with known subproblem structures,

A selection of tasks from public corpora such as SQuAD (for QA), CNN/DailyMail (for summarization), and arithmetic problem datasets,

A corpus of custom-designed complex prompts combining heterogeneous subtasks, specifically targeting the system's decomposition, orchestration, and aggregation capabilities,

Each test instance is annotated with the expected subtasks, ground truth results and metadata for performance comparison.

10.1.1 Illustrative Modular and Hormonal Workflows

[22]To complement the experimental protocol, we provide four illustrative scenarios that demonstrate the modularity and symbolic hormonal reactivity of the S-AI system. These examples correspond to the task types evaluated in the benchmark datasets and highlight how S-AI dynamically adapts to context.

1. Question Answering (QA) – with focus hormone

A natural language question is decomposed into three subtasks: question analysis, span localization, and answer extraction. The symbolic hormone focus_boost is injected to enhance comprehension under ambiguity.

```
FUNCTION run_sai_qa_pipeline_hormonal():
    hormonal_engine ← SimulatedHormonalEngine()
    question ← "What does AI simulate?"
    context ← "AI is a field that simulates human intelligence."
    hormonal_engine.inject_signal("focus_boost", level=0.8)
    subtasks ← [
        {task: "analyze_question", input: question},
        {task: "locate_span", input: context, keyword: "simulate"},
        {task: "extract_answer", input: context, anchor: "simulate"}
    ]
    FOR sub IN subtasks:
        IF sub.task == "analyze_question" AND hormonal_engine.get("focus_boost") > 0.5:
            DISPLAY "Better understanding enabled by hormonal boost"
        IF sub.task == "extract_answer":
            DISPLAY "Answer: human intelligence"
```

2. Text Summarization – with compression hormone

A short article is summarized by selecting key facts, extracting entities, and generating a final sentence. Hormone compression_boost adjusts the condensation level.

Generated summary: "A potentially habitable exoplanet has been discovered by scientists."

Hormonal modulation: If compression_boost > 0.8 → stricter selection of key facts.

3. Arithmetic Reasoning – with calculation focus

A verbal math problem is solved via modular steps: quantity extraction, relation detection, and numeric computation. Hormone calculation_focus increases logical discipline.

Computed answer: 180 km

Activated hormone: calculation_focus. If high → reinforced logic and formula verification: distance = speed × time.

4. Hybrid Multi-Tag Task – with contextual tag-based hormones

A task combining anomaly detection and summarization receives contextual tags like finance and urgent, which trigger dopamine and cortisol signals influencing how subtasks are prioritized.

Subtask 1: anomaly detection → rare events prioritized if cortisol is high.

Subtask 2: trend summarization → emphasis on positive evolution if dopamine is present.

These scenarios highlight the harmonious integration of semantic decomposition, sparse activation of specialized agents, and contextual hormonal modulation. Their modular and explainable execution exemplifies the core principles of the S-AI system as evaluated experimentally.

10.1.2 Classical Pipelines vs S-AI: A Comparative Perspective

To highlight the advantages of S-AI, Table 10.1 presents a direct comparison between the S-AI system and classical AI pipelines applied to the same four scenarios described in Section 10.1.1. These reference approaches include pre-trained transformer-based models, symbolic rule systems, and hybrid AI pipelines, all of which were used as baselines in our experimental protocol (see Section 10.3).

Each of the four tasks—question answering, text summarization, arithmetic reasoning, and hybrid multi-task processing—was executed both through S-AI and through a traditional pipeline. This comparison sheds light on key differences in modularity, explainability, computational frugality, and contextual adaptability.

Table 10.1 – Comparative Analysis of Classical Pipelines vs S-AI (by Task Type)

Task Type	Classical Pipeline	S-AI Approach
Question Answering (QA)	Monolithic Transformer (e.g., BERT, RoBERTa), no task decomposition	Modular subtasks (analyze → locate → extract), enhanced by focus_boost hormone
Text Summarization	End-to-end seq2seq model (e.g., T5, BART), no transparency or control	Structured extraction and synthesis, with adjustable condensation via compression_boost
Arithmetic Reasoning	Prompt-based LLM or rule engine; no explicit step-wise trace	Symbolic modular pipeline (extract → relate → compute) with logical regulation via calculation_focus

Task Type	Classical Pipeline	S-AI Approach
Hybrid Multi-Tag Task	Multi-task transformer or scripted rules, no contextual prioritization	Hormonal tag-driven prioritization (dopamine, cortisol) and modular agent selection

Unlike traditional pipelines that treat input holistically and rely on end-to-end implicit reasoning, S-AI explicitly models the problem-solving process as a combination of decomposed subtasks, selectively activated agents, and context-sensitive modulation. This leads to better transparency, dynamic resource management, and explainable behavior—key requirements for sustainable, trustworthy AI.

10.2 Evaluation Metrics

The evaluation relies on a combination of quantitative and qualitative measures, aligned with the core objectives of the S-AI system:

Evaluation Axis	Metrics Used
Decomposition Quality	Subproblem classification accuracy, rule coverage
Agent Performance	Task-specific precision (ROUGE, BLEU, MAE, etc.)
Orchestration Efficiency	Average number of agents activated per task
Computational Cost	CPU time, memory consumption, energy usage
Result Coherence	Human evaluation, semantic similarity scores
Explainability	Trace completeness, user satisfaction
Hormonal Reactivity	Adjusted response time, strategy modulation rate, context-based adaptation

These metrics enable a comprehensive evaluation of the system's modularity, parsimony, robustness, transparency, and contextual sensitivity.

10.3 Comparative Results

To contextualize S-AI's performance, we compared it with several reference systems:

A monolithic pre-trained transformer-based model ,

A rule-based system with no adaptive orchestration,

A neuro-symbolic hybrid system without explicit modular architecture.

Key findings include:

A significative reduction in average computational cost, particularly on tasks with easily identifiable simple subtasks,

Improved interpretability, with clear traceability of decomposition and activated agents,

Comparable or slightly superior accuracy metrics for decomposable tasks,

Better resilience to task heterogeneity due to the system's modular adaptation,

These results confirm the hypothesis that intelligent decomposition combined with parsimonious activation—and enhanced by hormonal regulation—yields an optimal trade-off between cost, quality, and adaptability.

10.4 Qualitative Analysis and Case Studies

Beyond numerical results, we conducted in-depth case studies to analyze the system's behavior on complex multi-step tasks. Among the examined examples:
A hybrid text-math problem requiring sequential reasoning,
A summarization task combining outputs from multiple agents,
A QA scenario with conflicting responses among sub-agents,
In each case, execution logs and agent activations were analyzed to highlight the modular coherence of reasoning and the added value of adaptive orchestration. The behavior of the Result Aggregator was also evaluated under different weighting modes revealing clear gains in robustness and response clarity.
These studies demonstrate that S-AI offers not only efficient performance but also adaptive, explainable, and human-readable reasoning traces, thereby enhancing user trust in the system.

Section 11 – Implementation of the S-AI System

In this section, we provide a detailed description of the implementation of the Sparse Artificial Intelligence (S-AI) system through pseudocode. Each critical component of the system is presented with its respective pseudocode and explanations, highlighting the functional flow of the system and how different parts interact with each other. The main components covered include the MetaAgent, the DecompositionAgent, the Specialized Agents, The Gland Agent, the Result Aggregator, and the main.py script that initializes and ties everything together.

11.1 MetaAgent as the Central Orchestrator

The MetaAgent plays a crucial role in coordinating the activities of specialized agents, handling hormonal signals, and aggregating results. Below is the pseudocode outlining the main tasks performed by the MetaAgent.

```
MetaAgent Initialization:
Initialize DQN model (DQNTrainer)
Initialize hormonal engine (HormonalEngine)
Load hormonal context (load_hormonal_context)
Initialize specialized agents (Math, Finance, Medical, ImageProcessing, TextAnalysis, Prediction)
Initialize result aggregator (ResultAggregator)
Set GlandAgent to None by default (can be assigned externally)
Emit Hormonal Signal:
Create a HormonalSignal with type, intensity, and target domain
Inject signal into hormonal engine
Update Hormonal Signals:
Decay all active hormonal signals
Remove inactive signals
Update hormonal context
Decompose Problem:
Check if problem has subproblems
If yes, return them, else return the problem as is
Classify Subproblem:
Estimate difficulty of the subproblem using DQN model
Adjust difficulty based on hormonal context (e.g., urgency)
```

```
Classify as "complex" or "simple" based on threshold
Select Agents:
Infer the domain of the subproblem
If domain is boosted by hormonal signal (e.g., attention), prioritize domain
Select the corresponding specialized agent based on the domain
Activate Agents:
For each selected agent:
Check if hormonal signals affect agent's behavior (e.g., fast_mode, defer_execution)
Execute agent in appropriate mode (fast, deferred, or normal)
Collect results from agent
Aggregate Results:
Use result aggregator to combine results from all agents
Log the final aggregated result
Learn from Feedback:
Adjust reward based on confidence bias from hormonal context
Update DQN model with feedback from subproblem outcome
Infer Domain:
Based on keywords in the subproblem, infer its domain (finance, medical, image, etc.)
Process Problem:
If a Gland Agent is assigned:
Request a hormonal profile based on the problem
Synthesize hormones from the Gland Agent
Inject synthesized hormones into the hormonal engine
Decompose the problem into subproblems
Select agents for each subproblem
Activate the selected agents
Aggregate all results into the final output
```

11.2 DecompositionAgent for Problem Breakdown

The DecompositionAgent is responsible for decomposing the complex problem into simpler subproblems using various strategies (symbolic, neural, or heuristic). Below is the pseudocode for the DecompositionAgent:

```
DecompositionAgent Initialization:
Load symbolic rule engine from RULE_CONFIG_PATH
Load neural classifier for complexity estimation
Store hormonal engine (HormonalEngine)
Store optional gland agent (GlandAgent)
Main Method: process(problem, hormonal_context)
Extract problem description
Initialize empty trace list
  If mode includes symbolic:
      - Apply symbolic rules to description with rule_engine
      - Append trace: method = symbolic_rules
      - If hormonal_context contains "rule_boost":
          - Multiply confidence by rule_boost modifier
      - If adjusted confidence >= symbolic threshold:
          - Return subproblems and trace
  If mode includes neural:
      - Predict complexity with neural classifier
      - Create generic subproblem: analyze_with_model
      - Append trace: method = neural_fallback_complex / simple
      - Inject hormone "mental_fatigue" into hormonal engine
      - Return subproblems and trace
  Heuristic Fallback:
      - Call decompose(problem)
      - Append trace: method = heuristic_fallback
      - Inject hormone "cognitive_overload" into hormonal engine
      - Return subproblems and trace
Method: decompose(problem)
  - Initialize empty subproblem list
  If problem is dict:
      For each key, value in problem:
          - If value is list or dict:
              → Add {key: value} to subproblems
          - If value is long text:
              → Add {"text": value}
              → Inject hormone "urgency"
          - If key contains "symptom":
              → Add {"symptoms": value}
              → Inject hormone "medical_attention"
          - If key contains "image" or "url":
              → Add {"image_url": value}
              → Inject hormone "high_load"
          - If key contains "data":
              → Add {"data": value}
          - Else:
              → Add {key: value}
  Else if problem is string:
      - Split into sentences
      - Add each non-empty sentence to subproblems
      - If > 3 sentences:
          → Inject hormone "urgency"
  Else:
      → Add [problem] as fallback
  - Return list of subproblems
Optional Method: update_hormonal_context_from_gland(problem)
  - If gland_agent is assigned:
```

```
    → Select hormonal profile
    → Synthesize hormones based on profile
    → Inject each hormone into hormonal engine
    → Log injected hormones.
```

11.3 GlandAgent for Hormonal Profile Generation

```
CLASS BaseGlandAgent:
  INIT(name)
  METHOD emit_hormones(agent_id, context):
    // Abstract method – must be implemented by each gland
CLASS AdrenalGland EXTENDS BaseGlandAgent:
  INIT(): name = "Adrenal"
  METHOD emit_hormones(agent_id, context):
    stress ← context.get("perceived_threat")
    cortisol ← compute intensity + duration based on stress
    profile ← {"gland:Adrenal": {"Cortisol": cortisol}}
    merge_hormonal_profile(agent_id, profile)
CLASS MesencephalonGland EXTENDS BaseGlandAgent:
  INIT(): name = "Mesencephalon"
  METHOD emit_hormones(agent_id, context):
    motivation ← context.get("goal_progress")
    dopamine ← compute intensity + duration
    store {"gland:Mesencephalon": {"Dopamine": dopamine}}
CLASS PinealGland EXTENDS BaseGlandAgent:
  INIT(): name = "Pineal"
  METHOD emit_hormones(agent_id, context):
    time ← context.get("time_of_day")
    melatonin ← compute intensity + duration
    store {"gland:Pineal": {"Melatonin": melatonin}}
CLASS ThyroidGland EXTENDS BaseGlandAgent:
  INIT(): name = "Thyroid"
  METHOD emit_hormones(agent_id, context):
    load ← context.get("processing_load")
    t3 ← intensity inversely proportional to load
    store {"gland:Thyroid": {"T3": t3}}
CLASS PituitaryGland EXTENDS BaseGlandAgent:
  INIT(): name = "Pituitary"
  METHOD emit_hormones(agent_id, context):
    novelty ← context.get("task_novelty")
    acth ← intensity = novelty
    store {"gland:Pituitary": {"ACTH": acth}}
CLASS PancreasGland EXTENDS BaseGlandAgent:
  INIT(): name = "Pancreas"
  METHOD emit_hormones(agent_id, context):
    density ← context.get("data_density")
    insulin ← compute intensity + duration
    store {"gland:Pancreas": {"Insulin": insulin}}
CLASS GlandSystem:
  INIT():
    glands ← list of all 6 gland instances
  METHOD emit_all(agent_id, context):
    FOR each gland IN glands:
      gland.emit_hormones(agent_id, context)
CLASSE BaseGlandAgent:
  INIT(nom)
  MÉTHODE emit_hormones(agent_id, contexte):
    // Méthode abstraite – chaque glande doit l'implémenter
CLASSE AdrenalGland HÉRITE BaseGlandAgent:
  INIT(): nom = "Adrenal"
  MÉTHODE emit_hormones(agent_id, contexte):
    stress ← contexte.get("perceived_threat")
    cortisol ← calculer intensité + durée selon le stress
    profil ← {"glande:Adrenal": {"Cortisol": cortisol}}
    fusionner_profil(agent_id, profil)
CLASSE MesencephalonGland HÉRITE BaseGlandAgent:
  INIT(): nom = "Mesencephalon"
  MÉTHODE emit_hormones(agent_id, contexte):
    motivation ← contexte.get("goal_progress")
    dopamine ← calculer intensité + durée
    stocker {"glande:Mesencephalon": {"Dopamine": dopamine}}
CLASSE PinealGland HÉRITE BaseGlandAgent:
  INIT(): nom = "Pineal"
  MÉTHODE emit_hormones(agent_id, contexte):
    heure ← contexte.get("time_of_day")
    melatonine ← calculer intensité + durée
    stocker {"glande:Pineal": {"Melatonin": melatonine}}
CLASSE ThyroidGland HÉRITE BaseGlandAgent:
  INIT(): nom = "Thyroid"
  MÉTHODE emit_hormones(agent_id, contexte):
    charge ← contexte.get("processing_load")
    t3 ← intensité inversement proportionnelle à la charge
    stocker {"glande:Thyroid": {"T3": t3}}
CLASSE PituitaryGland HÉRITE BaseGlandAgent:
  INIT(): nom = "Pituitary"
  MÉTHODE emit_hormones(agent_id, contexte):
    nouveauté ← contexte.get("task_novelty")
    acth ← intensité = nouveauté
    stocker {"glande:Pituitary": {"ACTH": acth}}
CLASSE PancreasGland HÉRITE BaseGlandAgent:
  INIT(): nom = "Pancreas"
  MÉTHODE emit_hormones(agent_id, contexte):
    densité ← contexte.get("data_density")
    insuline ← calculer intensité + durée
    stocker {"glande:Pancreas": {"Insulin": insuline}}
CLASSE GlandSystem:
  INIT():
    glands ← liste des 6 glandes
  MÉTHODE emit_all(agent_id, contexte):
    POUR chaque glande DANS glands:
      glande.emit_hormones(agent_id, contexte)
```

11.4 HormonalEngine for Behavioral Signal Routing

```
# engine/hormonal_engine.py
# Artificial Hormonal Signaling Engine + Symbolic Inference Layer for Sparse Artificial Intelligence (S-AI)
import numpy as np
import logging
import json
logger = logging.getLogger("HormonalEngine")
class HormonalEngine:
"""
  HormonalEngine simulates artificial hormonal signaling to regulate adaptive
  behaviors within the MetaAgent and Specialized Agents of the S-AI system.
  """
  def __init__(self, hormones=None):
    """
    Initialize the HormonalEngine.
    Args:
      hormones (dict): Optional dictionary of initial hormone levels.
    """
    self.hormones = hormones if hormones else {}
  def release_hormone(self, hormone, amount):
    """
    Release or increase the level of a specified hormone.
    Args:
      hormone (str): Name of the hormone to release.
      amount (float): Amount by which the hormone's level increases.
    """
    if hormone in self.hormones:
            self.hormones[hormone] += amount
        else:
            self.hormones[hormone] = amount
        logger.debug(f"Hormone '{hormone}' increased by {amount:.3f} →
{self.hormones[hormone]:.3f}")
    def degrade_hormones(self, decay_rate=0.1):
        """
        Gradually degrade hormone levels to simulate biological decay.
        Args:
            decay_rate (float): Proportional decay rate applied to all hormones.
        """
        for hormone in self.hormones:
            old_level = self.hormones[hormone]
            self.hormones[hormone] *= (1 - decay_rate)
            logger.debug(f"Hormone '{hormone}' decayed: {old_level:.3f} →
{self.hormones[hormone]:.3f}")
    def get_hormone_level(self, hormone):
        """
        Get the current level of a specified hormone.
        Args:
            hormone (str): Name of the hormone.
        Returns:
            float: Current level of the hormone, or 0 if hormone doesn't exist.
        """
        return self.hormones.get(hormone, 0.0)
    def hormonal_response(self, stimuli, sensitivity=1.0):
        """
        Adjust hormone levels based on external stimuli.
        Args:
            stimuli (dict): Dictionary of stimuli with corresponding intensity.
            sensitivity (float): Factor to modulate the hormone reaction.
        """
        for hormone, intensity in stimuli.items():
            delta = sensitivity * intensity
            self.release_hormone(hormone, delta)
            logger.debug(f"Stimulus → Hormone '{hormone}' adjusted by {delta:.3f}")
class HormonalInferenceEngine:
    """
    Symbolic rule-based inference engine operating on the current hormone levels
    provided by the HormonalEngine to trigger high-level behavioral states.
    """
    def __init__(self):
        self.rules = []
```

```
 def add_rule(self, condition_fn, consequence_label):
    """
    Add a new inference rule.
    Args:
       condition_fn: A lambda function receiving a hormone dict and returning bool
       consequence_label: A symbolic label (e.g., 'CRITICAL_MODE')
    """
    self.rules.append((condition_fn, consequence_label))
    logger.info(f"Symbolic rule added: '{consequence_label}'")
 def infer_modes(self, hormone_levels):
    """
    Apply all inference rules to the current hormone levels.
    Args:
       hormone_levels (dict): Current hormone levels from HormonalEngine
    Returns:
       list of str: All matched symbolic consequences.
    """
    consequences = []
    for condition_fn, label in self.rules:
       try:
          if condition_fn(hormone_levels):
             consequences.append(label)
           logger.debug(f"Inference matched: {label}")
       except Exception as e:
logger.warning(f"Error evaluating rule '{label}': {e}")
    return consequences
 def load_rules_from_json(self, filepath):
    """
    Load symbolic inference rules from a JSON file.
    Each rule must contain 'label' and 'condition' as a string expression.
    Example rule:
    {
       "label": "CRITICAL_MODE",
   "condition": "urgency > 0.8 and confidence < 0.5"
    }
    """
    try:
       with open(filepath, 'r') as f:
          rules_data = json.load(f)
       for rule in rules_data:
          label = rule.get("label")
          condition_expr = rule.get("condition")
          if not label or not condition_expr:
```

```
          logger.warning(f"Invalid rule skipped: {rule}")
                continue
        # Unsafe eval (for now): assumes trusted source
            condition_fn = eval(f"lambda h: {condition_expr}")
            self.add_rule(condition_fn, label)
        logger.info(f"{len(rules_data)} rules loaded from {filepath}")
    except Exception as e:
  logger.error(f"Failed to load rules from {filepath}: {e}")
```

11.5 Hormonal Context and Agent Behavior Modulation

The hormonal context plays a crucial role in modulating agent behavior, influencing decisions such as the speed of execution, the selection of agents, and the prioritization of results. Below is the pseudocode for handling hormonal signals:

plaintext

Copier

```
Function ProcessHormonalSignals(hormonal_context):
    # Step 1: Evaluate hormonal influence on behavior
    If "urgency" in hormonal_context:
        Adjust execution speed or agent selection based on urgency
    If "high_load" in hormonal_context:
Defer execution for certain agents if system load is high
    Return modified results based on hormonal context
```

Explanation:

Urgency Signal: This signal accelerates the processing of certain subproblems that are considered urgent, such as financial transactions or critical medical cases.

High Load Signal: This signal slows down or defers the execution of agents if the system is under high load, ensuring that resources are not overburdened.

11.6 MedicalAgent – A Hormone-Aware Specialized Agent# agents/medical_agent.pyxxx

As a representative example of a Specialized Agent in the S-AI system, we provide below the pseudocode of the MedicalAgent. This agent is designed to process subproblems that describe medical symptoms in natural language, using a transformer-based NLP model (DistilBERT). It also demonstrates how hormonal context—such as urgency, stress, or low confidence—can influence its reasoning, response format, and execution mode. The MedicalAgent illustrates the modularity, adaptability, and context sensitivity that characterize all agents in the S-AI architecture.

```
import logging
from transformers import pipeline
from agents.agent_base_agent import BaseAgent
logger = logging.getLogger("MedicalAgent")
class MedicalAgent(BaseAgent):
    def __init__(self, name="medical"):
        self.name = name
        logger.info("Chargement du modèle NLP pour l'agent médical")
        self.model = pipeline("text-classification", model="distilbert-base-uncased")

    def execute(self, symptoms, results=None, fast=False, hormonal_context=None):
        logger.info("Activation de l'agent Médical")
        try:
            if not isinstance(symptoms, str) or len(symptoms) < 5:
                # ✅ Amélioration 1 : redirection ou reformulation si entrée trop courte
                return {
                    "error": "Données invalides. Fournissez une description plus détaillée des symptômes.",
                    "suggestion": "Reformulez la demande ou redirigez vers TextAnalysisAgent."
                }
            # --- Analyse du contexte hormonal ---
            urgency = hormonal_context.get("urgency", 0) if hormonal_context else 0
            stress = hormonal_context.get("stress", 0) if hormonal_context else 0
            low_conf = hormonal_context.get("low_confidence", 0) if hormonal_context else 0
            # Mode rapide forcé si urgence détectée
            if urgency > 0.5 or fast:
                symptoms = symptoms[:64]
# traitement accéléré : tronquage
            result = self.model(symptoms)
            # Ajustement selon stress
            if stress > 0.7:
                result.append({"note": "⚠ Contexte stressé détecté. Résultat à interpréter prudemment."})
            # Ajustement selon niveau de confiance
            if low_conf > 0.5:
                result.append({"note": "ℹ Ce diagnostic automatique est à titre indicatif uniquement."})

            # ✅ Amélioration 2 : enrichissement avec métadonnées
            metadata = {
                "origin": "MedicalAgent",
                "hormones": {
                    "urgency": urgency,
                    "stress": stress,
                    "low_confidence": low_conf
                },
                "warnings": [item["note"] for item in result if "note" in item]
            }
            return {
                "result": result,
                "metadata": metadata
            }
        except Exception as e:
            logger.error(f"Erreur dans MedicalAgent: {e}")
            return {"error": str(e)}
    def process(self, subproblem, fast=False, hormonal_context=None):
        logger.info("Traitement du sous-problème par MedicalAgent")
        try:
            if isinstance(subproblem, dict) and 'symptoms' in subproblem:
                return self.execute(subproblem['symptoms'], fast=fast, hormonal_context=hormonal_context)
            elif isinstance(subproblem, str):
                return self.execute(subproblem, fast=fast, hormonal_context=hormonal_context)
            else:
                return {"error": "Format de sous-problème non reconnu pour l'agent médical."}
        except Exception as e:
            logger.error(f"Erreur dans process() de MedicalAgent: {e}")
            return {"error": str(e)}
```

11.7 main.py – Central Orchestration and Execution Pipelin

The main.py file serves as the entry point for the S-AI system. It is responsible for initializing all components, connecting to the database, configuring the cache, and orchestrating the interaction between agents and the MetaAgent. It also initializes the Gland Agents, which are responsible for synthesizing hormonal profiles based on the selected behavioral context. Below is the pseudocode for the main.py file:

```
FUNCTION main():
  LOG "Starting the S-AI system"
  // Step 1: Connect to database
  db ← connect_mysql()
  IF db IS NULL:
      LOG "Error: Failed to connect to database"
      RETURN
  // Step 2: Initialize core components
  cache ← RedisCache()
  config ← Config()
  hormonal_engine ← HormonalEngine()
  // Step 3: Define a problem
  problem ← {
      type: "finance",
      description: "Suspicious transaction analysis",
      data: [...]
  }
  // Step 4: Define contextual triggers for glandular hormones
  context ← {
      perceived_threat, goal_progress,
      time_of_day, processing_load,
      task_novelty, data_density
  }
  // Step 5: Generate hormonal profile via Gland Agents
```

```
    gland_system ← GlandSystem()
    gland_system.emit_all(agent_id = problem.type, context = context)

    // Step 6: Inject optional symbolic hormonal signals
    IF problem.type == "finance":
        hormonal_engine.inject_signal("urgency")
    ELSE IF problem.type == "image":
        hormonal_engine.inject_signal("high_load")
    // Step 7: Initialize decomposition and meta agents
    decomposition_agent ← DecompositionAgent(config, hormonal_engine)
    meta_agent ← MetaAgent(config, cache, db)
    meta_agent.hormonal_engine ← hormonal_engine
    // Step 8: Process the problem
    result ← meta_agent.process_problem(problem)
    // Step 9: Display final result
    PRINT result
    // Step 10: Display symbolic hormone state
    PRINT hormonal_engine.get_current_hormones()
    // Step 11: Clear hormonal engine
    hormonal_engine.clear()
//  Optional hybrid mode entry point
FUNCTION run_hybrid_tasks():
    tasks ← load_custom_tasks()
    hormonal_engine ← HormonalEngine()
    decomposition_agent ← DecompositionAgent()
    meta_agent ← MetaAgent()
    meta_agent.hormonal_engine ← hormonal_engine
    FOR EACH task IN tasks:
        // Inject hormones from tags
        FOR EACH tag IN task.context_tags:
            FOR EACH (hormone, level) IN tag_to_hormone[tag]:
                hormonal_engine.release(hormone, level)

        // Decompose and process task
        subproblems ← decomposition_agent.decompose(task.description, task.subtasks)
        result ← meta_agent.handle_subproblems(subproblems)
        // Output
        PRINT "  Result:", result
        PRINT "  Expected:", task.expected_output
        PRINT "  Active Hormones:", hormonal_engine.get_current_hormones()
        // Reset hormonal engine
        hormonal_engine.reset()
//  CLI-based entry point
```

```
IF __name__ == "__main__":
    mode ← parse_cli_arg("--mode", default="standard")
    IF mode == "standard":
        main()
    ELSE IF mode == "hybrid":
        run_hybrid_tasks()
```

11.8 Project Directory Structure

The architecture of the S-AI system is organized in a modular way to facilitate scalability and maintainability. Below is the complete directory structure of the project:

```
project_S_AI/
│
├── models/
│   ├── __init__.py
│   ├── logger.py
│   ├── cache.py
│   ├── config.py
│   ├── Dqn.py
│   └── rule-engine.py
│
├── agents/
│   ├── __init__.py
│   ├── agent_base_agent.py      ← (nommé "mobaquent" par convention)
│   ├── decomposition_agent.py
│   ├── specialized_agent.py
│   └── meta_agent_hormonal.py
│
├── engine/
│   ├── __init__.py
│   ├── hormonal_engine.py
│   └── hormonal_context.py
│
├── aggregator/
│   ├── __init__.py
│   └── result_aggregator.py
│
├── knowledge_base/
│   ├── __init__.py
│   └── knowledge_base.py
│
├── database/
│   ├── __init__.py
│   └── database.py
│
├── experiments/
│   ├── __init__.py
│   ├── evaluation_tests.py
│   ├── sai_custom_tasks.jsonl
│   └── load_custom_tasks.py
│
├── utils/
│   ├── __init__.py
│   └── helpers.py
│
├── docs/
│   ├── diagrams/
│   │   └── architecture_UML.png
│   ├── S_AI_article_EN.md
│   ├── S_AI_article_FR.md
│   └── README.md
│
├── tests/
│   ├── __init__.py
│   ├── unit_tests.py
│   ├── test-decomposition-agent.py
│   ├── test_meta_agent.py
│   └── test_hormonal_agent.py
│
├── requirements.txt
├── setup.py
├── README.md
└── main.py
```

Section 12 – Perspectives and Future Work

12.1 Multi-Agent Cooperation and Task Parallelization

One of the most promising extensions of the S-AI framework is the integration of cooperative agents capable of solving subproblems in parallel while sharing their intermediate knowledge. This would enable more efficient handling of large-scale problems through task distribution and inter-agent communication. Future versions of the system could incorporate advanced coordination strategies, such as agent negotiation, dependency resolution, or shared memory models, to enhance the synergy between modules and reduce execution times.

12.2 Towards a Globally Self-Evolving Architecture

Section 5 introduced the initial foundations of sparse meta-learning, integrated into both the Decomposition Agent and the MetaAgent. A major future direction involves extending this adaptive learning dynamic to the entire S-AI architecture, including activation mechanisms, evaluation protocols, and aggregation strategies.

Rather than relying on fixed strategies, each system component could progressively refine its internal rules based on observed results, errors, and feedback.

This shift from local adaptability to systemic self-organization would pave the way for an AI system capable of intelligently reconfiguring itself in response to new contexts, domains, or constraints, while preserving the core principles of parsimony, modularity, and transparency.

A promising direction is the co-adaptation between the MetaAgent and the Gland Agents. These endocrine-inspired components could learn to refine hormonal profiles based on feedback from previous activations. By observing the system's global performance under different hormonal contexts, Gland Agents could adjust their synthesis strategies to better match future tasks. This would enable a form of meta-hormonal learning, reinforcing the synergy between top-down orchestration and decentralized modulation.

12.3 Dynamic Agent Generation and Modular Plug-in System

S-AI could integrate a more flexible agent management mechanism, enabling the dynamic generation of new agents or the modular integration of pre-trained plug-ins.

Instead of relying on a static agent library, the MetaAgent could dynamically invoke external APIs, LLM models, or specialized expert agents as needed. This would transform S-AI into a self-extensible and adaptable system, adjusting its architecture based on operational constraints and runtime context.

12.4 Deployment on Embedded and Edge Systems

Thanks to its parsimonious and modular design, S-AI is well-suited for deployment in resource-constrained environments such as embedded systems, IoT devices, or mobile platforms.

A lightweight S-AI configuration can act as a local inference core, while heavier agents can be executed remotely via asynchronous orchestration in the cloud. This opens practical perspectives in fields like autonomous robotics, smart monitoring, or ambient contextual intelligence.

12.5 Towards Real-World Deployment of S-AI

Beyond architectural design and experimental validation, transitioning Sparse Artificial Intelligence (S-AI) into real-world environments raises several practical challenges and unlocks concrete opportunities.

The initial deployment process is structured as follows:

Instantiation of the MetaAgent and Decomposition Agent,

Registration of available Specialized Agents (SAs),

Definition of orchestration rules and hormonal sensitivity profiles.

These steps can be automated through configuration files or dynamic discovery protocols. Modular libraries allow dynamic loading of agents and long-term performance monitoring.

In production environments, S-AI components can be deployed as microservices or containers (Docker, Kubernetes), facilitating integration into existing software architectures. REST or GraphQL APIs allow tasks to be submitted, results retrieved, or system states queried in real time.

Supervision, hormonal adaptability, and modular scalability are essential. Specialized Agents can be updated or replaced incrementally without disrupting the overall system. Monitoring dashboards provide visibility over agent activations, confidence levels, active hormonal signals, and aggregated performance.

This operationalization paves the way for resource-efficient, context-aware AI engines, well-suited to the constraints of real-world intelligent systems.

12.6 Philosophical and Societal Perspectives

Beyond technical considerations, S-AI invites reflection on the future of AI design principles. By favoring minimalism, transparency, and modularity over brute-force complexity, it embodies a vision of sustainable and human-aligned intelligence. Its compositional logic mirrors human reasoning patterns and provides a more traceable and ethical alternative to opaque black-box systems. Future research should explore how S-AI architectures could foster trust, explainability, and democratic control over intelligent systems.

Section 13– Conclusion

Over the past decade, the field of artificial intelligence has been driven primarily by massive models, end-to-end training pipelines, and brute-force learning strategies. While these approaches have achieved impressive technical milestones, they have also led to systems that are increasingly opaque, resource-hungry, and difficult to adapt or explain.

This paper has introduced a fundamentally different paradigm: Sparse Artificial Intelligence (S-AI), which embraces modularity, selective activation, and intelligent orchestration as core design principles. By decomposing problems, activating only the necessary agents, and integrating results through lightweight coordination mechanisms, S-AI demonstrates that efficiency and explainability need not be sacrificed for performance.

In addition to these foundational principles, we proposed a biologically inspired extension: a layer of artificial hormonal signaling that provides indirect, context-sensitive regulation across the architecture. This signaling mechanism introduces a new dimension of distributed adaptability, enabling agents to adjust their behavior reactively without explicit instruction—much like hormones do in biological systems. It reinforces the parsimony, resilience, and systemic coherence of S-AI, especially in dynamic or ambiguous environments.

Throughout this work, we have presented:

A theoretical foundation grounded in activation parsimony,

A modular architecture including Decomposition Agent, MetaAgent, Specialized Agents, and Result Aggregator,

An operational layer of hormonal modulation for indirect inter-agent regulation,

Code implementations and orchestration strategies aligned with frugal AI principles,

Experimental validation showing reduced computational cost, robust task decomposition, and enhanced transparency.

More than just a technical solution, S-AI represents a philosophical stance: a call for a sustainable, ethical, and human-aligned AI. As future systems grow in scale and complexity, we believe that the S-AI principles—now extended with artificial endocrine signaling—will become increasingly relevant, offering a path toward intelligent systems that are not only powerful, but also transparent, adaptable, and responsible.

References

1. AI Agents Authors. (2025). AI Agents: Evolution, Architecture, and Real-World Applications. arXiv preprint arXiv:2503.12687.
2. Arora, D., Sonwane, A., Wadhwa, N., Mehrotra, A., Utpala, S., Bairi, R., Kanade, A., & Natarajan, N. (2024). MASAI: Modular Architecture for Software-engineering AI Agents. arXiv preprint arXiv:2406.11638.

3. DARA Authors. (2024). DARA: Decomposition-Alignment-Reasoning Autonomous Language Agent for Question Answering over Knowledge Graphs. arXiv preprint arXiv:2402.05982.
4. DIEM Authors. (2024). DIEM: Decomposition-Integration Enhancing Multimodal Insights. arXiv preprint arXiv:2403.02679.
5. Durfee, E. H. (2001). Distributed Problem Solving and Planning. In Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence (pp. 121–164). MIT Press.
6. Lesser, V. (1999). Cooperative Multiagent Systems: A Personal View of the State of the Art. IEEE Transactions on Knowledge and Data Engineering, 11(1), 133–142.
7. Li, D., Tan, Z., Qian, P., Li, Y., Chaudhary, K. S., Hu, L., & Shen, J. (2024). SMoA: Improving Multi-agent Large Language Models with Sparse Mixture-of-Agents. arXiv preprint arXiv:2411.03284.
8. MoA Authors. (2024). SMoA: Improving Multi-agent Large Language Models with Sparse Mixture-of-Agents. arXiv preprint arXiv:2411.03284.
9. MoE Pruning Authors. (2024). A Provably Effective Method for Pruning Experts in Fine-tuned Sparse Mixture-of-Experts. arXiv preprint arXiv:2402.08196.
10. Neural Decomposition Authors. (2024). Unveiling Options with Neural Network Decomposition. ICLR Conference. OpenReview: https://openreview.net/forum?id=MCjU9KcPBL
11. Orthogonal Experts Authors. (2024). Multi-Task Reinforcement Learning with Mixture of Orthogonal Experts. ICLR Conference. OpenReview: https://openreview.net/forum?id=O1z3gSk0PR
12. Parametric Subtasks Authors. (2023). Parameterizing Non-Parametric Meta-Reinforcement Learning Tasks via Subtask Decomposition. arXiv preprint arXiv:2310.00248.
13. Ravuru, C., Sakhinana, S. S., & Runkana, V. (2024). Agentic Retrieval-Augmented Generation for Time Series Analysis. arXiv preprint arXiv:2408.14484.
14. Russell, S., & Norvig, P. (2021). Artificial Intelligence: A Modern Approach (4th ed.). Pearson Education.
15. Slaoui, S., et al. (2023). Achievements of Artificial Intelligence in the Past and During the COVID-19 Era to Tackle Deadly Diseases. Mechanisms and Machine Science, 121, 181–191.
16. Slaoui, S., et al. (2022). An Efficient Parallel Algorithm for Clustering Big Data based on the Spark Framework. International Journal of Advanced Computer Science and Applications, 13(7), 890–896.
17. Slaoui, S., et al. (2021). A Parameter-free Clustering Algorithm based on K-means. International Journal of Advanced Computer Science and Applications, 12(3), 612–619.
18. Slaoui, S., et al. (2021). A survey on parallel clustering algorithms for Big Data. Artificial Intelligence Review, 54(4), 2411–2443.
19. Slaoui, S., et al. (2018). E-Transitive: An enhanced version of the transitive heuristic for clustering categorical data. Procedia Computer Science, 127, 26–34.
20. Slaoui, S., et al. (2018). Pdc-Transitive: An enhanced heuristic for document clustering based on relational analysis approach and iterative MapReduce. Journal of Information and Knowledge Management, 17(2), 1850021.
21. Slaoui, S., et al. (2017). Clustering categorical data based on the relational analysis approach and MapReduce. Journal of Big Data, 4(1), 28.
22. Slaoui, S., et al. (2016). Parallel document clustering using iterative MapReduce. ACM International Conference Proceeding Series, a37.
23. Slaoui, S., et al. (2015). Clustering of large data based on the relational analysis. In 2015 Intelligent Systems and Computer Vision (ISCV 2015), 7105550.

24. Sun, C., Shen, M., & How, J. P. (2020). Scaling Up Multiagent Reinforcement Learning for Robotic Systems: Learn an Adaptive Sparse Communication Graph. arXiv preprint arXiv:2003.01040.
25. TDAG Authors. (2024). TDAG: A Multi-Agent Framework based on Dynamic Task Decomposition and Agent Generation. GitHub Repository: https://github.com/AGI-Edgerunners/LLM-Agents-Papers
26. Vision-Language MoE Authors. (2024). Boosting Continual Learning of Vision-Language Models via Mixture-of-Experts Adapters. arXiv preprint arXiv:2401.12399.
27. Wooldridge, M. (2009). An Introduction to MultiAgent Systems (2nd ed.). Wiley.
28. Foerster, J. N., Assael, Y. M., de Freitas, N., & Whiteson, S. (2016). Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In Advances in Neural Information Processing Systems (NeurIPS), 29. https://arxiv.org/abs/1605.06676
29. Zhang, K., Yang, Z., & Basar, T. (2021). Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. Handbook of Reinforcement Learning and Control, 321–384. https://arxiv.org/abs/1911.10635
30. Kim, J., Song, J., & Kim, K. (2019). Learning to Schedule Communication in Multi-Agent Reinforcement Learning. In International Conference on Learning Representations (ICLR). https://openreview.net/forum?id=H1xwNhC9tQ