BRAINAGENT-CGP: CONSTRAINT-GUIDED HIERARCHICAL PLANNING FOR RELIABLE AUTONOMOUS BRAIN-SIGNAL ANALYSIS

Anonymous authors
Paper under double-blind review

ABSTRACT

Electroencephalography (EEG) analysis often relies on expert-designed pipelines in which processing steps, tool order, and intermediate data are defined manually. Large language model (LLM)-based agents can automate parts of this process from natural-language requests, but their generated workflows may still be invalid when required preprocessing steps or intermediate artifacts are missing. We propose BrainAgent-CGP (Constraint-Guided Planning), a multi-agent framework that separates semantic goal interpretation from executable workflow construction. Each analytical tool declares its required and produced artifacts, supported goals, agent ownership, execution cost, and reliability. A uniform-cost planner uses this information to build dependency-valid workflows, followed by deterministic agent routing, runtime validation, and state-preserving replanning. BrainAgent-CGP is evaluated on 30 natural-language tasks spanning low, medium, and high workflow complexity. It achieves a Planning Success rate, Plan Validity Rate, Required-Tool Recall, and routing accuracy of 1.00, with an Invalid Tool Rate of 0.00. In the ablation study, removing artifact constraints reduces Plan Validity Rate to 0.233 and increases Invalid Tool Rate to 0.273, showing the importance of explicit dependency modeling. A language-robustness study over 90 paraphrased and unseen requests achieves an overall goal exact-match rate of 0.989 with the hybrid semantic interpreter while maintaining a Plan Validity Rate of 1.00. Controlled failure experiments achieve a Recovery Rate of 1.00 when an alternative path is available and a Safe Deferral Rate of 1.00 when no valid alternative exists. Finally, BrainAgent-CGP successfully executes 48 tool operations across 12 ISRUC-Sleep and Sleep-EDF Expanded recordings totaling 177.41 h of EEG/PSG data. These results show that explicit artifact-state planning can improve the reliability of autonomous brain-signal analysis.

1 INTRODUCTION

Electroencephalography is a high-temporal-resolution, non-invasive modality used in sleep research, cognitive monitoring, affective computing, neurological assessment, and brain-computer interfaces (Zhang et al., 2025a; Kumar & Sharma, 2025). Traditionally, EEG analysis has relied on expert-designed processing pipelines implemented using tools such as MATLAB, EEGLAB, MNE-Python, or custom software (Falach et al., 2025; Gramfort et al., 2013; Delorme & Makeig, 2004). The analyst manually defines the required steps, their order, and the intermediate data products needed by downstream analyses. This approach is effective but requires substantial domain and programming expertise. Recent advances in large language models (LLMs) provide an opportunity to automate parts of this process through natural-language interaction.

LLM-based agents can interpret analytical goals, select tools, and construct multi-step workflows (Jiang et al., 2025). EEG Agent (Zhao et al., 2026) uses an LLM-enabled framework to plan and schedule EEG analysis tools, while BrainAgent (Zhou et al., 2026) introduces a hierarchical supervisor and specialized sub-agents for autonomous brain-signal analysis. These systems reduce the need for users to manually define every analysis step.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

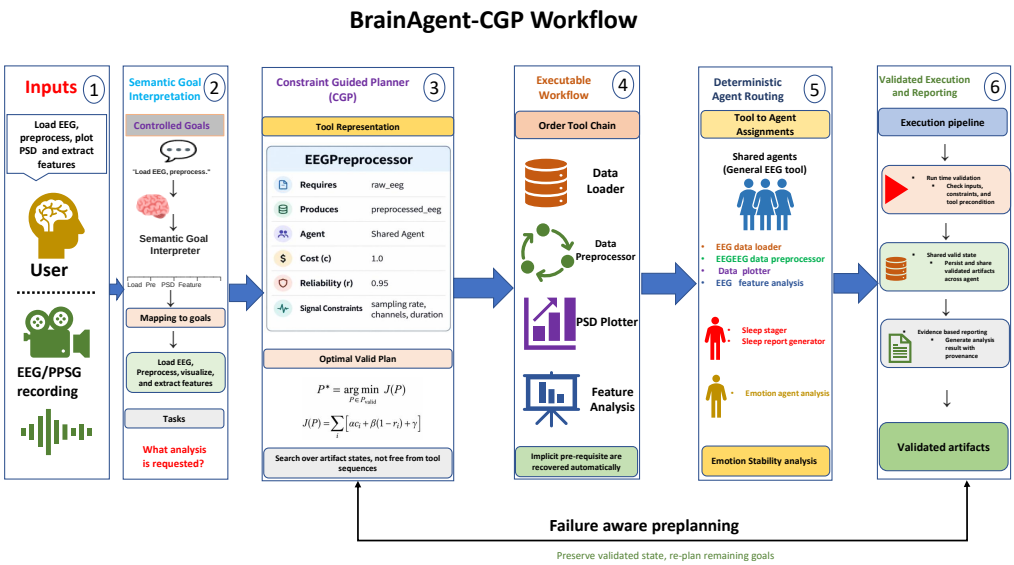


Figure 1: BrainAgent-CGP separates semantic intent interpretation from constraint-valid workflow construction and validated execution.

However, an LLM-generated workflow may be semantically relevant but still computationally invalid. A selected tool may require preprocessing or intermediate artifacts that have not yet been produced. For example, plotting EEG power spectral density may require the signal to be loaded and preprocessed before the PSD tool can run. Existing LLM-enabled brain-signal agents may therefore omit necessary prerequisite steps, creating a gap between semantic relevance and executable workflow construction.

This problem has also motivated broader research on structured planning and explicit tool dependencies in LLM agents. Plan-over-Graph organizes subtasks as graphs (Liu et al., 2026), Graph Co-Pilot models explicit API dependencies (Liu & Zou, 2026), GTool performs graph-enhanced tool planning (Chen et al., 2026), GraphPlan represents prerequisite- and outcome-conditioned transitions (Li et al.), and HyperAgent models tools as schema-transforming hyperedges (Zhai et al., 2026). These studies show the value of structured planning for reliable tool use. Brain-signal analysis, however, introduces domain-specific requirements such as preprocessing dependencies, intermediate signal states, channel availability, and agent-specific responsibilities. This work therefore focuses on constraint-guided planning for autonomous brain-signal analysis.

To address this problem, we propose BrainAgent-CGP, a constraint-guided multi-agent framework for reliable autonomous brain-signal analysis. BrainAgent-CGP separates semantic goal interpretation from executable workflow construction. Natural-language requests are mapped to supported analytical goals, while a constraint-guided planner constructs a valid tool sequence based on required and produced artifacts, execution cost, and reliability. The framework also supports deterministic agent routing, runtime validation, and state-preserving replanning after execution failures, as illustrated in Fig. 1. The main contributions are summarized below:

- We propose BrainAgent-CGP, a constraint-guided multi-agent framework that separates semantic interpretation from executable planning and enforces artifact dependencies before execution.

- We formulate brain-signal workflow construction as a typed artifact-state planning problem in which each tool defines its required and produced artifacts, analytical goals, agent ownership, execution cost, and reliability.

- We develop a Constraint-Guided Planner based on uniform-cost state-space search. It expands only tools whose prerequisites are satisfied and recovers implicit dependencies that may be omitted from the user request.
- We separate semantic goal interpretation, workflow construction, and agent routing while supporting runtime validation and state-preserving replanning. Tools are routed deterministically to the Shared, Sleep, or Emotion agents.
- We evaluate BrainAgent-CGP using a 30-task planning benchmark, 90 paraphrased and unseen requests, controlled failure and ablation experiments, and 12 real ISRUC-Sleep and Sleep-EDF Expanded recordings. The planner achieves complete validity and required-tool coverage on the 30-task benchmark, while all 48 real-data tool executions complete successfully across 177.41 h of EEG/PSG data.

2 Related Work

In the literature, it has been reported that EEG analysis traditionally relies on multistage pipelines that combine signal preprocessing with temporal, spectral, spatial, nonlinear, or connectivity features, followed by statistical learning or task-specific machine-learning models. Deep learning algorithms have reduced the need for manually designed features because they learn signal representations directly from EEG. A compact example is EEGNet, a convolutional neural network architecture designed for EEG decoding. This pretrained model has substantially advanced EEG decoding. However, it assumes that the analytical task and processing pipeline are specified in advance. Therefore, they address how a given EEG analysis should be performed rather than how an open-ended user request should be translated into a valid sequence of analytical operations.

In recent years, large language models have been used to design more flexible prototypes for EEG analysis. Zhao et al. (2026) proposed EEGAgent, which uses an LLM to schedule external tools for EEG information extraction, spatiotemporal analysis, event detection, interaction, and report generation. This approach shifts EEG analysis from a fixed task-specific pipeline toward natural-language-driven tool orchestration. BrainAgent (Zhou et al., 2026) extends this direction through a hierarchical multi-agent architecture consisting of a central supervisor, shared EEG-processing tools, and domain-specific agents for sleep and emotion analysis. It also introduces a benchmark covering atomic, sequential, and cross-domain requests.

Despite these advances, workflow construction in existing brain-signal agents remains dependent on language-model reasoning over tool descriptions. A selected tool may be relevant to the requested analysis but still fail because a required preprocessing step, signal representation, channel, or intermediate output is unavailable. Zhou et al. (2026) exemplified this by reporting an alpha-band visualization task that fails because the required preprocessing operation is omitted from the generated workflow. This example shows that selecting a relevant tool does not necessarily produce an executable workflow. Therefore, semantic interpretation and workflow executability need to be considered separately.

This limitation is also related to a broader problem in LLM-based tool-using agents. Several existing systems select tools sequentially from natural-language descriptions, while dependencies among tools remain implicit. Zhang et al. (2025b) addressed this issue by introducing structured representations of tasks, tools, inputs, and outputs. Liu et al. (2026) proposed Plan-over-Graph, which represents decomposed tasks as graphs and generates schedules based on their dependencies. The main focus of this framework is task decomposition and parallel execution rather than domain-specific data validity. Chen et al. (2026) proposed GTool, which constructs request-specific tool graphs and shows that treating tools independently can lead to invalid plans. Similarly, Wu et al. (Wu et al., 2025) modeled inter-task dependencies to determine which operations can execute in parallel and which must remain sequential.

In recent studies, different approaches have been proposed to incorporate explicit execution requirements into the planning process. Xia et al. (Ming et al., 2025) proposed GraSP, which represents skills as typed directed acyclic graphs with precondition-effect relations and supports verification and localized repair during execution. Zhai et al. (Zhai et al., 2026) proposed HyperAgent, which represents tools using input and output schemas and resolves missing requirements during execution when additional information is needed. These studies show that explicit dependency modeling can

improve the reliability of tool-based agents. However, these approaches are mainly designed for general-purpose tasks and do not explicitly consider the physiological data dependencies required for brain-signal analysis.

The existing brain-signal agents mainly use LLM reasoning to select and organize analytical tools, while recent general-purpose planning frameworks introduce graphs, tool dependencies, preconditions, or runtime repair mechanisms. BrainAgent-CGP builds on these directions but focuses specifically on executable brain-signal workflows. It separates semantic interpretation from workflow construction and uses explicit artifact dependencies, deterministic agent routing, runtime validation, and state-preserving replanning.

3 BRAINAGENT-CGP

The existing LLM-enabled brain-signal agents are able to interpret natural-language requests and select relevant analytical tools. However, the reliable execution of these tools remains challenging because brain-signal workflows often contain implicit dependencies that are not stated in the user request most of the time. For example, spectral analysis requires preprocessing; a sleep report requires sleep-stage outputs, and emotion-related analysis depends on specific channels or a processed EEG signal. Likewise, even a valid workflow may fail because of missing channels, unsupported signal conditions, unavailable tools, or invalid intermediate outputs.

In our proposed BrainAgent-CGP, we address these challenges by separating semantic interpretation from executable workflow construction. Given a natural-language request and an EEG or PSG recording, our framework first identifies the required analytical goals. The Constraint-Guided Planner then selects the tools and determines their execution order based on explicit artifact dependencies, execution cost, and reliability. Unlike existing LLM-enabled EEG agents, the executable tool sequence is not determined directly by language-model reasoning. A tool can execute only when its required artifacts are available. The selected tools are routed to the corresponding Shared, Sleep, or Emotion agent, and each intermediate result is validated before it is used by downstream operations. If a tool fails, BrainAgent-CGP preserves the previously validated results and replans the remaining workflow.

3.1 ARTIFACT-CONSTRAINED ANALYTICAL STATE

To support constraint-guided planning, BrainAgent-CGP uses the user request, analytical tools, and intermediate outputs in a structured analytical state. This formulation provides a formal representation of how a user request is translated into executable analytical goals.

Let Q denote a natural-language analytical request. The semantic layer maps Q to a controlled goal set, which is generalized below.

$$G_Q = \mathrm{Parse}(Q), \qquad G_Q \subseteq \mathcal{G}, \tag{1}$$

In Eq. 1, $\mathcal{G}$ denotes the predefined analytical goal vocabulary, where example goals include EEG loading, preprocessing, spectral visualization, feature extraction, sleep staging, sleep-report generation, emotion-stability analysis, and mental-workload analysis. After identifying the requested analytical goals, the planner determines which registered tools can satisfy these goals and which dependencies must be satisfied before each tool can execute. The semantic layer can use either controlled phrase matching or a constrained LLM-based mapper, while workflow construction remains the responsibility of the planner. We represent each analytical tool as follows.

$$t_i = \langle I_i, O_i, G_i, A_i, c_i, r_i, \Pi_i \rangle, \tag{2}$$

In Eq. 2, I_i denotes the set of required input artifacts, while O_i denotes the set of artifacts produced by the tool. G_i represents the analytical goals supported by the tool, A_i identifies its owning agent, $c_i \geq 0$ denotes the execution cost, and $r_i \in [0, 1]$ represents the reliability estimate. The term Π_i denotes optional signal-level constraints such as required channels, feasible sampling rate, or minimum recording duration. Representative tool-instantiation examples are provided in Appendix D.

BrainAgent-CGP maintains the analytical state, formalized as follows:

$$s_k = (V_k, H_k, F_k), \tag{3}$$

In Eq. 3, V_k contains the artifacts that have already been generated and validated. The term H_k records the execution history, while F_k contains tools that have failed or are currently unavailable. Typical elements of V_k include `raw_eeg`, `processed_eeg`, `quality_report`, `sleep_stages`, `spectral_features`, and `emotion_biomarkers`. Using the state defined in Eq. 3, a tool t_i is executable only when all of its requirements are satisfied:

$$t_i \in \mathcal{T}(s_k) \iff I_i \subseteq V_k \wedge \Pi_i(V_k) = 1 \wedge t_i \notin F_k. \tag{4}$$

In Eq. 4, the condition $I_i \subseteq V_k$ ensures that all required input artifacts are available in the current state. The condition $\Pi_i(V_k) = 1$ indicates that the signal-level requirements of the tool are satisfied. The condition $t_i \notin F_k$ ensures that a failed or unavailable tool is not selected for execution. When the tool executes successfully, its output artifacts are added to the validated state, as follows:

$$V_{k+1} = V_k \cup O_i. \tag{5}$$

In Eq. 5, O_i denotes the set of artifacts produced by tool t_i. These artifacts become available to subsequent tools and may satisfy the input requirements of later analytical operations. This representation makes the analytical dependencies explicit. A downstream tool can execute only when its required artifacts are available and the tool has not previously failed. Therefore, BrainAgent-CGP builds the workflow progressively from validated intermediate states rather than generating an unconstrained sequence of tools.

3.2 Constraint-Guided Planning

The previous subsection introduced the artifact-state representation. Given this foundation, the Constraint-Guided Planner (CGP) searches for an executable sequence of tools that satisfies the analytical goals derived from the user request. Unlike an LLM-enabled approach that directly predicts the complete tool order, CGP constructs the workflow from the current validated state. In our framework, at each planning step, only tools whose required artifacts are available are considered for expansion. The current implementation uses uniform-cost search over the reachable artifact states. To guide this search, each executable tool t_i is assigned an incremental planning cost, which is generalized below.

$$J(t_i) = c_i + \lambda(1 - r_i). \tag{6}$$

In Eq. 6, c_i denotes the registered execution cost of tool t_i, while r_i represents its reliability estimate. The parameter $\lambda \geq 0$ controls the penalty assigned to less reliable tools. A candidate workflow containing L tools is represented as follows.

$$P = (t_1, t_2, \ldots, t_L), \tag{7}$$

The total planning cost is obtained by accumulating the individual tool costs:

$$J(P) = \sum_{i=1}^{L} \left[c_i + \lambda(1 - r_i) \right]. \tag{8}$$

In Eq. 8, workflows containing higher-cost or less reliable tools receive a higher planning cost. Therefore, the planner favors a lower-cost valid workflow rather than simply selecting every tool that may be relevant to the request. The workflow-selection objective is defined as below.

$$P^* = \arg \min_{P \in \mathcal{P}_{\text{valid}}} J(P), \tag{9}$$

In Eq. 9, $\mathcal{P}_{\text{valid}}$ denotes the set of workflows that satisfy the planning constraints. A candidate workflow is valid only when the requested goals are achieved and every selected tool is executable from the state available before its execution. These conditions are expressed as

$$\begin{aligned} &G_Q \subseteq \Gamma(V_P), \\ &I_i \subseteq V_{i-1},\ \Pi_i(V_{i-1}) = 1,\ t_i \notin F_{i-1}, \quad \forall\, t_i \in P. \end{aligned} \tag{10}$$

In Eq. 10, $\Gamma(V_P)$ denotes the analytical goals satisfied by the final validated artifact state V_P. The conditions in Eq. 10 require each selected tool to have its inputs, satisfy its signal constraints, and remain available before execution. These constraints prevent the planner from constructing a workflow with missing dependencies or invalid tool transitions. Because the incremental costs in Eq. 6 are non-negative, uniform-cost search expands candidate states in increasing order of accumulated cost. Therefore, among the reachable valid workflows, the first goal-satisfying workflow removed from the priority queue has the lowest planning cost.

Algorithm 1 summarizes the CGP workflow. The search begins from the initial validated artifact set V_0. Each search entry contains the accumulated cost, current artifact state, and partial workflow. The planner repeatedly expands the lowest-cost state and evaluates the registered tools against the current artifact requirements. A valid tool produces a new artifact state, which is inserted into the search queue when a lower-cost path to that state has been found.

The complete pseudocode and illustrative dependency-recovery examples are provided in Appendix D.

3.3 Execution, Validation, and Reporting

Once CGP returns a valid workflow, each tool is routed according to the agent ownership defined in the tool registry:

$$\text{Route}(t_i) = A_i. \tag{11}$$

In Eq. 11, A_i denotes the agent responsible for executing tool t_i. General EEG operations, such as loading, preprocessing, quality assessment, PSD analysis, and feature extraction, are handled by the shared analytical layer. Sleep-related operations are routed to `SleepAgent`, while emotion- and workload-related operations are routed to `EmotionAgent`. Because agent ownership is predefined in the tool registry, no additional language-model decision is required for routing.

During execution, each tool result is checked before the workflow continues. The executor verifies that the tool returns a valid result and that all artifacts declared by the tool are present in the shared state. If either check fails, execution raises an error and the corresponding tool is treated as failed.

Detailed output-validation and state-preserving replanning equations are provided in Appendix D.

4 Experimental Setup

The complete tool suite, dataset descriptions, experimental protocol, and metric definitions are provided in Appendices A–C. The main text retains the experimental results needed to evaluate the proposed framework.

5 Experimental Results

We evaluate BrainAgent-CGP through complementary planning, language, execution, failure-recovery, and ablation experiments. We first examine real-data execution across two EEG/PSG datasets, followed by the expanded 30-task planning benchmark and repeated comparison with three LLM-based planners. We then evaluate robustness to paraphrased and unseen requests, failure-aware replanning, task complexity, and the contribution of the main planning components.

5.1 Cross-Dataset Real-Data Execution

We evaluate the same artifact-state workflow on six ISRUC-Sleep and six Sleep-EDF Expanded recordings using the request "Load EEG, preprocess it, extract frequency features, and plot PSD." Across both datasets, BrainAgent-CGP processes 638,662 s (177.41 h) of real EEG/PSG data. All 12 workflows and all 48 tool executions complete successfully with no execution failures. The same analytical goals, artifact dependencies, and tool registry are used for both datasets. Detailed per-recording results, the aggregate execution table, dependency path, and PSD outputs are provided in Appendix F.

5.2 Expanded Planning Benchmark

BrainAgent-CGP generates dependency-valid workflows for all 30 tasks, achieving Planning Success, Plan Validity Rate, Required-Tool Recall, and routing accuracy of 1.00, with an Invalid Tool Rate of 0.00 and no redundant tool selections. Mean workflow length increases from 2.40 tools for low- complexity tasks to 6.40 for high-complexity tasks, while mean planning latency increases from 0.0004 s to 0.1110 s. Complete complexity-level results are provided in Appendix E.

5.3 Repeated 30-Task Planner Comparison

We compare BrainAgent-CGP with three LLM-based planning strategies using the same analytical tool registry and 30-task benchmark. The baselines include a Single-LLM planner, a hierarchical supervisor/sub-agent planner, and a ReAct-style iterative planner. Each LLM-based method is evaluated over 10 runs per task, resulting in 300 evaluations per baseline. BrainAgent-CGP is deterministic and is evaluated once on each task.

Because this experiment evaluates workflow construction rather than end-to-end signal execution, Planning Success (PS) is reported instead of Task Completion Rate (TCR). Table 1 summarizes planning quality, workflow efficiency, and planning latency.

Table 1: Planner comparison on the 30-task benchmark. Each LLM-based planner is run 10 times per task, for 300 evaluations per method.

Method	PS ↑	PVR ↑	RTR ↑	Route ↑	ITR ↓	Latency (s) ↓	Tools ↓	Redundant ↓	Cost ↓
Single LLM	1.000	0.933	0.988	1.000	0.008	2.507	6.00	1.95	7.79
Hierarchical LLM	1.000	0.930	0.991	1.000	0.019	5.973	9.04	4.93	11.19
ReAct-style	0.923	0.847	0.990	0.923	0.015	7.595	4.75	1.05	6.12
BrainAgent-CGP	1.000	1.000	1.000	1.000	0.000	0.038	4.13	0.00	5.46

BrainAgent-CGP maintains a Planning Success rate, Plan Validity Rate, Required-Tool Recall, and routing accuracy of 1.00, with an Invalid Tool Rate of 0.00 and no redundant tool selections. The Single-LLM and Hierarchical-LLM planners achieve complete Planning Success but show lower plan validity and higher redundancy. The ReAct-style planner also shows lower Planning Success, Plan Validity Rate, and routing accuracy. BrainAgent-CGP also has the lowest mean planning latency at 0.038 s, compared with 2.507 s for the Single LLM, 5.973 s for the Hierarchical LLM, and 7.595 s for the ReAct-style planner. It selects an average of 4.13 tools with no redundant selections. These results show that explicit artifact-state planning maintains valid workflows while reducing redundant tool selection and planning latency.

5.4 Robustness to Paraphrased and Unseen Natural-Language Requests

We evaluate whether the semantic interpretation layer can recover the intended analytical goals when users express the same tasks using different natural-language formulations. The original 30-task benchmark is expanded with three variants per task, producing 90 additional requests. The hybrid semantic interpreter successfully parses all 90 requests and achieves an overall goal exact-match rate of 0.989, goal recall of 0.996, and goal precision of 1.000. Planning Success and Plan Validity Rate remain 1.00, routing accuracy remains 1.00, and the Invalid Tool Rate remains 0.00. Detailed controlled-parser results and the complete robustness table are provided in Appendix G.

These results support the separation between semantic interpretation and executable planning. The semantic front end can be improved without changing the artifact-state planner, while CGP continues to recover dependencies and construct valid workflows.

5.5 Failure-Aware Replanning

We evaluate whether BrainAgent-CGP preserves validated state and responds safely when tool execution fails. In all three native no-alternative cases, the failed tool is identified, previously validated artifacts are preserved, and the framework safely defers. In all three controlled recovery cases, BrainAgent-CGP blacklists the failed tool, replans from the preserved artifact state, selects the fallback producer, and completes the requested goal.

The Safe Deferral Rate and Recovery Rate are both 1.00, no previously successful tool is re-executed, and the Artifact Preservation Rate and Failed-Tool Identification Rate are both 1.00. The complete fault-injection protocol and results are provided in Appendix H.

5.6 Ablation Study

We perform an ablation study to examine the contribution of the main planning components in BrainAgent-CGP. All variants use the same 30-task benchmark, semantic parser, and tool registry. We evaluate three variants: removing artifact constraints, removing the reliability penalty, and replacing the cost-and-reliability objective with unit-cost search. Table 2 shows that artifact constraints have the largest effect on planning reliability. Without them, Planning Success decreases from 1.000 to 0.767, Plan Validity Rate (PVR) from 1.000 to 0.233, and Required-Tool Recall (RTR) from 1.000 to 0.717, while Invalid Tool Rate (ITR) increases from 0.000 to 0.273. The lower workflow length and TTC in this variant do not indicate better efficiency because many generated workflows are incomplete or invalid.

Table 2: Planning-level ablation study on the 30-task benchmark.PS denotes Planning Success, PVR Plan Validity Rate, RTR Required-Tool Recall, RA routing accuracy, ITR Invalid Tool Rate, and TTC mean registered Total Tool Cost.

Method	PS	PVR	RTR	RA	ITR	Tools	Redundant	TTC
Full BrainAgent-CGP	1.000	1.000	1.000	1.000	0.000	4.13	0.00	5.46
w/o Artifact Constraints	0.767	0.233	0.717	0.767	0.273	2.40	0.00	3.13
w/o Reliability Penalty	1.000	1.000	1.000	1.000	0.000	4.13	0.00	5.46
w/o Cost + Reliability	1.000	1.000	1.000	1.000	0.000	4.13	0.00	5.46

The complexity-specific artifact-constraint ablation is provided in Appendix I.

Removing the reliability penalty or replacing the complete objective with unit-cost search does not change the selected workflows on the current benchmark. This is expected because the current production registry has a single producer for each artifact, so artifact dependencies largely determine the feasible workflow. Cost and reliability are expected to have a greater effect when multiple valid tools can produce the same artifact.

6 Discussion

The results show that BrainAgent-CGP can construct dependency-valid brain-signal workflows by separating semantic interpretation from executable planning. Across the 30-task benchmark, BrainAgent-CGP achieves complete Planning Success, Plan Validity Rate, Required-Tool Recall, and routing accuracy, with no invalid or redundant tool selections. Increasing task complexity mainly affects workflow length and planning latency rather than planning correctness.

The main advantage of BrainAgent-CGP is its explicit artifact-state representation. A tool may be relevant to a user request but still be invalid to execute when its required artifacts are unavailable. The ablation study confirms this distinction: removing artifact constraints reduces PVR from 1.00 to 0.233 and increases ITR from 0.00 to 0.273, with the strongest degradation observed for high-complexity tasks. Separating semantic interpretation from workflow construction also improves language flexibility. The hybrid semantic interpreter increases overall goal exact match to 0.989 on 90 paraphrased and unseen requests, while the same CGP planner maintains a PVR of 1.00, routing

accuracy of 1.00, and ITR of 0.00. This shows that the semantic front end can be improved without changing the dependency rules used for executable planning.

The real-data and failure experiments provide complementary evidence beyond planner-only evaluation. BrainAgent-CGP completes all 48 tool executions across 12 ISRUC-Sleep and Sleep-EDF Expanded recordings totaling 177.41 h of EEG/PSG data. Under controlled tool failures, the framework preserves previously validated state, recovers when an alternative producer is available, and safely defers when no valid alternative remains. These results support explicit artifact-state planning as a practical approach for reliable autonomous brain-signal workflow construction.

7 LIMITATIONS

BrainAgent-CGP has some limitations. First, the real-data evaluation includes 12 recordings from two datasets and evaluates one canonical multi-step workflow. Additional datasets, recordings, and workflow types are needed to assess broader generalizability.

Second, planner behavior depends on the correctness of the tool registry. Required and produced artifacts, agent ownership, execution cost, and reliability are currently specified manually. Incorrect registry entries may therefore affect workflow construction.

Third, the current sleep-staging method is used only for software and workflow testing. The evaluated datasets also do not provide suitable emotion or workload ground truth; therefore, these outputs are treated as signal-derived research descriptors rather than predictive labels.

BrainAgent-CGP is intended as a research framework for brain-signal analysis and is not designed for clinical diagnosis.

8 CONCLUSION

In this work, we presented BrainAgent-CGP, a constraint-guided framework for constructing executable brain-signal analysis workflows from natural-language requests. The framework separates semantic goal interpretation from executable workflow construction and explicitly models required and produced artifacts, agent ownership, execution cost, and tool reliability. Across the expanded 30-task benchmark, BrainAgent-CGP achieves Planning Success, Plan Validity Rate, Required-Tool Recall, and routing accuracy of 1.00, with an Invalid Tool Rate of 0.00 and no redundant tool selections. The results remain consistent across low-, medium-, and high-complexity tasks, showing that the artifact-state planner maintains valid workflow construction as task complexity increases. The real-data evaluation further shows that the same workflow rules operate across different recording configurations. BrainAgent-CGP completes all 12 workflows and all 48 tool executions on six ISRUC-Sleep and six Sleep-EDF Expanded recordings totaling 177.41 h of EEG/PSG data. Repeated comparison with Single-LLM, Hierarchical-LLM, and ReAct-style planners shows that BrainAgent-CGP maintains valid and compact workflows with lower planning latency. The language-robustness experiment also shows that the semantic front end can be improved without changing the constraint-guided planner. Controlled failure experiments further show that BrainAgent-CGP preserves validated state, achieves a Recovery Rate of 1.00 when an alternative path is available, and safely defers all no-alternative cases without repeating previously successful operations. Future work will extend the framework to larger tool registries, additional datasets and workflow types, and dynamically estimated tool reliability.

ETHICS STATEMENT

This study uses publicly available EEG/PSG datasets and does not involve new human-subject data collection. The current framework is intended for research use and is not designed for clinical diagnosis. Signal-derived emotion, workload, and sleep-related outputs are treated as research descriptors unless appropriate ground truth is available.

REPRODUCIBILITY STATEMENT

The main planning formulation and evaluation results are described in the paper. The complete tool suite, dataset and protocol details, metric definitions, secondary analyses, and additional experimental results are provided in the appendix. We will share the code and experiment-specific configuration files upon acceptance.

AI USE STATEMENT

Generative AI tools were used for language editing and code-development assistance. All AI-assisted manuscript edits and code were reviewed and verified by the authors. The authors take responsibility for the final content, claims, experiments, and artifacts presented in this work.

REFERENCES

D. Alvarez-Estevez and R. Rijsman. Haaglanden medisch centrum sleep staging database, 2022. PhysioNet, version 1.1.

W. Chen, D. Yao, W. Li, X. Meng, C. Gong, and J. Bi. Gtool: Graph enhanced tool planning with large language model. In *Proceedings of the International Conference on Learning Representations (ICLR)*, pp. 69247–69269, 2026.

A. Delorme and S. Makeig. Eeglab: An open source toolbox for analysis of single-trial eeg dynamics including independent component analysis. *Journal of Neuroscience Methods*, 134(1):9–21, 2004.

R. Falach, G. Belonosov, F. J. Schmidig, M. Aderka, V. Zhelezniakov, R. Shani-Hershkovich, et al. Sleepeegpy: A python-based software integration package to organize preprocessing, analysis, and visualization of sleep eeg data. *Computers in Biology and Medicine*, 192:110232, 2025.

A. Gramfort, M. Luessi, E. Larson, D. A. Engemann, D. Strohmeier, C. Brodbeck, et al. Meg and eeg data analysis with mne-python. *Frontiers in Neuroinformatics*, 7:267, 2013.

T. Jiang, J. Wu, and S. C. Leung. The cognitive impacts of large language model interactions on problem solving and decision making using eeg analysis. *Frontiers in Computational Neuroscience*, 19:1556483, 2025.

S. Khalighi, T. Sousa, J. M. Santos, and U. Nunes. Isruc-sleep: A comprehensive public dataset for sleep researchers, 2026. NEMAR, version 1.0.1.

S. Kumar and A. Sharma. Advances in non-invasive eeg-based brain-computer interfaces: Signal acquisition, processing, emerging approaches, and applications. In *Signal Processing Strategies*, pp. 281–310. 2025.

C. Li, S. Cheng, Y. Zhang, J. Li, Y. Liu, and W. Huang. Graphplan: Graph-enhanced planning via thinking llms for embodied agents. Reference metadata as provided in the original manuscript.

G. Liu and X. Zou. Graph co-pilot: Explicit api dependency modeling for reliable multi-tool llm reasoning. In *Proceedings of the 6th International Conference on Neural Networks, Information and Communication Engineering (NNICE)*, pp. 867–871, 2026.

Y. Liu, G. Zhang, K. Wang, S. Li, S. Pan, and B. An. Graph-augmented large language model agents: Current progress and future prospects. *IEEE Intelligent Systems*, 41(2):45–55, 2026.

Z. Ming, Y. Huang, M. Liu, J. Zhang, S. Liu, and T. Yan. Grasp what you see: Toward fine-grained brain-controlled robotic arm manipulation in 3d iot scenarios. *IEEE Transactions on Mobile Computing*, 2025.

T. Pollard, B. E. Moody, L.-W. H. Lehman, B. J. Gow, C. Fernandes, C. Xie, A. Johnson, R. G. Mark, and T. Heldt. Physionet as a global platform for biomedical research. *Nature Health*, 1(8): 792–795, 2026. doi: 10.1038/s44360-026-00096-z.

Y. Wu, E. L. M. Su, M. Wu, C. Y. Ooi, and W. Holderbaum. A review of machine learning and deep learning trends in eeg-based epileptic seizure prediction. *IEEE Access*, 2025.

Z. Zhai, X. Tan, G. Zou, X. Wang, and W. Zhang. Hyperagent: Planning and acting over tool-schema hypergraphs for tool-use llm agents, 2026.

L. Zhang, B. Li, M. Cao, C. Peng, and H. Wang. Classification of eeg-fnirs bimodal brain signals for motor imagery tasks based on wavelet transform and spatio-temporal domain processing. *Neuroscience*, 2025a.

S. Zhang, X. Ma, Z. Cao, Z. Zhang, and H. Zhao. Plan-over-graph: Towards parallelable llm agent schedule, 2025b.

S. Zhao, M. Peng, H. Jiang, T. Li, and S. Li. Eeg agent: A unified framework for automated eeg analysis using large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18063–18071, 2026.

Y. Zhou, S. Zhao, J. Wang, S. Li, and G. Pan. Brainagent: A large language model-driven multi-agent framework for autonomous brain signal understanding, 2026.

APPENDIX

A COMPLETE TOOL SUITE

Our BrainAgent-CGP organizes its analytical tools into a shared EEG tool set and two domain-specific agents: `SleepAgent` and `EmotionAgent`.

Shared EEG tools. The shared tool set includes `EEGFileLoader`, `EEGPreprocessor`, `EEGSaver`, EEG plotting and PSD tools, `EEGFeatureAnalysis`, and `EEGQualityAssessor`. `EEGFileLoader` reads the input EEG or PSG recording and produces the initial `raw_eeg` artifact together with its metadata. `EEGPreprocessor` applies the requested signal-processing operations and produces the `processed_eeg` artifact required by downstream spectral and emotion-related analyses. The remaining shared tools support signal quality assessment, visualization, saving, and feature extraction.

SleepAgent. `SleepAgent` includes tools for sleep staging, hypnogram generation, sleep-report generation, spindle detection, K-complex detection, slow-wave detection, arousal analysis, and time-frequency analysis. Sleep-related dependencies are explicitly defined in the tool registry. For example, `SleepReportGenerator` requires a `sleep_stages` artifact and therefore cannot execute before sleep staging has been completed. When ground-truth sleep annotations are available, they are used as reference stage information. The spectral-rule staging method in the current implementation is used only for software testing and not for evaluating sleep-staging performance.

EmotionAgent. `EmotionAgent` includes tools for frontal alpha asymmetry, spectral biomarker analysis, temporal stability analysis, and workload- or fatigue-related EEG descriptors. The current spectral measures include beta-to-alpha, theta-to-beta, and theta-to-alpha ratios, together with alpha peak frequency and temporal variability. Small denominator floors are applied when spectral power approaches zero to avoid unstable ratio estimates. These outputs are treated as signal-derived research descriptors rather than clinical or affective labels. Because the sleep datasets used in the current experiments do not provide emotion or workload ground truth, these tools are used to evaluate workflow construction and execution rather than affective classification performance.

B DATASETS AND EXPERIMENTAL PROTOCOL

We evaluate BrainAgent-CGP using public EEG/PSG datasets with different recording configurations. ISRUC-Sleep (Khalighi et al., 2026) is used for development and initial real-data evaluation. The six evaluated ISRUC recordings contain 19 channels sampled at 200 Hz. We additionally evaluate BrainAgent-CGP on Sleep-EDF Expanded (Pollard et al., 2026), where the six evaluated recordings contain seven recorded signals sampled at 100 Hz. The two datasets differ in channel configuration, sampling rate, and recording duration, providing a cross-dataset setting for testing the artifact-state execution model. The framework also includes an adapter for the HMC Sleep Staging Database (Alvarez-Estevez & Rijsman, 2022); however, HMC is not included in the results reported in this work.

Benchmark design. The planning benchmark contains 30 natural-language brain-signal analysis requests, with 10 tasks at each of three complexity levels. Level 1 contains short workflows involving basic EEG operations, including loading, preprocessing, quality assessment, visualization, feature extraction, sleep analysis, and emotion-related analysis. Level 2 contains sequential multi-step requests with implicit analytical dependencies. Level 3 contains longer multi-goal requests that may require multiple analytical branches and coordination across the Shared, Sleep, and Emotion agents.

To evaluate robustness to linguistic variation, each benchmark task is also expressed using a light paraphrase, a conversational paraphrase, and an unseen formulation, producing 90 additional requests while preserving the original analytical intent and reference workflow. These requests are used to compare the controlled parser with the hybrid semantic interpreter. Each benchmark task specifies the natural-language request, target goals, required tools, expected routing, and complexity level. Reference workflows are defined by required tools and artifact dependencies rather than a single fixed tool order, allowing independent operations to execute in any valid order.

Baselines. All planning methods use the same analytical tool registry so that performance differences reflect planning behavior rather than tool availability. The evaluated baselines include a Single-LLM planner, a hierarchical supervisor/sub-agent planner, and a ReAct-style iterative planner. Each LLM-based planner is evaluated over 10 runs per task, while BrainAgent-CGP uses deterministic planning.

C Evaluation Metrics

We evaluate BrainAgent-CGP using complementary metrics that measure task completion, workflow validity, dependency satisfaction, routing correctness, semantic interpretation, and recovery from execution failures. Task Completion Rate (TCR) measures the fraction of evaluation requests for which the complete analytical task is successfully executed:

$$\mathrm{TCR} = \frac{N_{\mathrm{completed}}}{N_{\mathrm{total}}}. \tag{12}$$

In Eq. 12, $N_{\mathrm{completed}}$ is the number of requests that reach all requested analytical goals, while N_{total} is the total number of evaluated requests. Plan Validity Rate (PVR) measures the fraction of generated workflows in which all required artifact dependencies are satisfied:

$$\mathrm{PVR} = \frac{N_{\text{valid plans}}}{N_{\text{generated plans}}}. \tag{13}$$

In Eq. 13, a plan is considered valid when every tool is scheduled only after its required artifacts are available. Required-Tool Recall (RTR) measures whether the generated workflow contains the tools required by the reference task:

$$\mathrm{RTR} = \frac{|T_{\mathrm{executed}} \cap T_{\mathrm{required}}|}{|T_{\mathrm{required}}|}. \tag{14}$$

In Eq. 14, T_{required} denotes the reference set of tools needed to complete the task, while T_{executed} denotes the tools included in the generated workflow. Invalid Tool Rate (ITR) measures the fraction of tool invocations performed before their required artifacts are available:

$$\mathrm{ITR} = \frac{N_{\text{invalid invocations}}}{N_{\text{tool invocations}}}. \tag{15}$$

For controlled execution failures, Recovery Rate (RR) measures the fraction of recoverable failure cases that successfully reach the requested goals after replanning:

$$\mathrm{RR} = \frac{N_{\mathrm{recovered}}}{N_{\text{recoverable failure cases}}}. \tag{16}$$

We also report Safe Deferral Rate (SDR), which measures the fraction of no-alternative failure cases that terminate without executing an invalid downstream workflow. Artifact Preservation Rate measures whether previously validated artifacts remain available after a failure, while Failed-Tool Identification Rate measures whether the failed operation is correctly recorded during replanning.

For the language-robustness experiment, we report goal exact match, goal precision, goal recall, and goal F1 score by comparing the interpreted analytical goals with the reference goal set.

We additionally report routing accuracy, planning latency, the number of selected tools, redundant tool invocations, and total registered tool cost. For a workflow P, the Total Tool Cost (TTC) is defined as

$$\mathrm{TTC}(P) = \sum_{t_i \in P} c_i, \tag{17}$$

In Eq. 17, c_i denotes the registered execution cost of tool t_i. Lower TTC indicates a lower aggregate registered tool cost. For the LLM-based planning baselines, we use GPT-5.6 Luna (`gpt-5.6-luna`). Each LLM-based planner is evaluated over 10 runs per task on the 30-task benchmark, while BrainAgent-CGP uses deterministic planning. The hybrid semantic interpreter is evaluated on the 90 paraphrased and unseen language variants.

D Additional Details of Proposed Framework

For example, an EEG preprocessing tool can be represented as

$$t_{\text{pre}} = \big\langle \{\texttt{raw_eeg}\}, \{\texttt{processed_eeg}\}, \{\texttt{preprocess_eeg}\}, \texttt{Shared}, c_{\text{pre}}, r_{\text{pre}}, \Pi_{\text{pre}} \big\rangle. \tag{18}$$

In Eq. 18, the preprocessing tool requires a `raw_eeg` artifact and produces a `processed_eeg` artifact. The tool supports the `preprocess_eeg` goal and is assigned to the Shared agent. Its cost, reliability, and constraints are represented by c_{pre}, r_{pre}, and Π_{pre}, respectively. A sleep-report generation tool is generalized in a similar way:

$$t_{\text{report}} = \big\langle \{\texttt{sleep_stages}\}, \{\texttt{sleep_report}\}, \{\texttt{sleep_report}\}, \texttt{SleepAgent}, c_{\text{report}}, r_{\text{report}}, \Pi_{\text{report}} \big\rangle. \tag{19}$$

In Eq. 19, the tool requires the `sleep_stages` artifact and produces the `sleep_report` artifact. This dependency prevents the planner from generating a sleep report before sleep staging has been completed. The tool is assigned to the SleepAgent. Its cost, reliability, and constraints are represented by c_{report}, r_{report}, and Π_{report}, respectively.

Algorithm 1 Constraint-Guided Workflow Planning

Require: Requested goals G_Q, tool registry $\mathcal{T}$, initial artifacts V_0, unavailable tools F
Ensure: Executable workflow P^* or FAILURE

OPEN $\leftarrow \{(0, V_0, \emptyset)\}$
BEST$[V_0] \leftarrow 0$
while OPEN $\neq \emptyset$ **do**
  $(C, V, P) \leftarrow$ lowest-cost entry in OPEN
  Remove (C, V, P) from OPEN
  if $G_Q \subseteq \Gamma(V)$ **then**
    return P
  end if
  for each tool $t_i \in \mathcal{T}$ **do**
    if $t_i \notin F$ **then**
      if $I_i \subseteq V$ **and** $\Pi_i(V) = 1$ **then**
        $V' \leftarrow V \cup O_i$
        $C' \leftarrow C + J(t_i)$
        if $V' \notin$ BEST **or** $C' <$ BEST$[V']$ **then**
          BEST$[V'] \leftarrow C'$
          Insert $(C', V', P \oplus t_i)$ into OPEN
        end if
      end if
    end if
  end for
end while
return FAILURE

The artifact constraints also allow CGP to recover analytical steps that are not explicitly stated in the user request. For example, consider the request "analyze emotional stability and generate a sleep report." The request specifies the desired analyses but does not explicitly mention preprocessing or sleep staging.

The `EmotionStabilityAnalyzer` requires a `processed_eeg` artifact. Therefore, the planner must first load and preprocess the EEG signal before emotional-stability analysis can be performed. This dependency produces the following path:

$$\begin{aligned}\texttt{EEGFileLoader} &\rightarrow \texttt{EEGPreprocessor} \\ &\rightarrow \texttt{EmotionStabilityAnalyzer}.\end{aligned} \tag{20}$$

Similarly, the `SleepReportGenerator` requires a `sleep_stages` artifact. Therefore, the planner introduces the sleep staging operation before report generation:

$$\begin{aligned}\texttt{EEGFileLoader} &\rightarrow \texttt{SleepStager} \\ &\rightarrow \texttt{SleepReportGenerator}.\end{aligned} \tag{21}$$

Eqs. 20 and 21 illustrate how CGP recovers dependencies that are implicit in the user request. Therefore, the operations that are not directly mentioned by the user will be added when their outputs are required to reach the requested analytical goals. This allows BrainAgent-CGP to construct dependency-valid workflows rather than relying only on the sequence implied by the natural-language request.

$$\begin{aligned}\mathcal{V}(o_i) = V_{\text{schema}}(o_i) \wedge V_{\text{signal}}(o_i) \\ \wedge V_{\text{dependency}}(o_i) \wedge V_{\text{quality}}(o_i).\end{aligned} \tag{22}$$

In Eq. 22, schema validation checks whether the output follows the expected structure. Signal validation checks properties such as channel availability and sampling conditions. Dependency validation confirms that the output was produced from a valid analytical state, while quality validation rejects unusable intermediate results. If validation fails, the output artifact is not added to the validated state. The corresponding tool is instead added to the unavailable-tool set:

$$F_{k+1} = F_k \cup \{t_{\text{failed}}\}. \tag{23}$$

In Eq. 23, the failed tool is recorded so that it is not selected again during replanning. CGP then replans from the current validated state:

$$P' = \text{CGP}\left(G_{\text{remaining}}, V_k, F_{k+1}\right). \tag{24}$$

In Eq. 24, $G_{\text{remaining}}$ denotes the analytical goals that have not yet been satisfied. Because replanning starts from the current artifact state, previously completed results can be reused instead of repeating the entire workflow. If no valid path remains, the system returns an explicit failure or deferral. After execution, the validated analytical outputs are combined with relevant domain evidence for report generation.

E Planning Benchmark Details

E.1 Additional Details of Planning Benchmark

We evaluate BrainAgent-CGP on 30 natural-language analysis requests, including 10 low-, 10 medium-, and 10 high-complexity tasks. This experiment evaluates workflow construction rather than end-to-end signal execution; therefore, Planning Success (PS) should not be interpreted as Task Completion Rate (TCR). Table 3 summarizes the results.

Table 3: BrainAgent-CGP planning performance on the 30-task benchmark.

Level	Tasks	PS	PVR	RTR	Route	ITR	Latency (s)	Tools
Low	10	1.00	1.00	1.00	1.00	0.00	0.0004	2.40
Medium	10	1.00	1.00	1.00	1.00	0.00	0.0029	3.60
High	10	1.00	1.00	1.00	1.00	0.00	0.1110	6.40
Overall	30	1.00	1.00	1.00	1.00	0.00	0.0381	4.13

BrainAgent-CGP generates dependency-valid workflows for all 30 tasks, achieving PS, PVR, Required-Tool Recall (RTR), and routing accuracy of 1.00, with an Invalid Tool Rate (ITR) of 0.00. No redundant tools are selected. The mean workflow length increases from 2.40 tools for Level 1 to 3.60 for Level 2 and 6.40 for Level 3. Planning latency also increases with workflow complexity. Mean latency is 0.0004 s for Level 1, 0.0029 s for Level 2, and 0.1110 s for Level 3, with an overall mean of 0.0381 s. These results show that the artifact-state planner maintains valid workflow construction across short, sequential, and cross-agent requests.

F Detailed Cross-Dataset Execution

F.1 Cross-Dataset Execution Setup

We evaluate whether BrainAgent-CGP can execute the same analytical goals on EEG/PSG recordings with different recording configurations. The experiment includes six ISRUC-Sleep recordings and six Sleep-EDF Expanded recordings. The same natural-language request is used for every recording without dataset-specific planning rules:

> "Load EEG, preprocess it, extract frequency features, and plot PSD."

This request requires four analytical operations. The recording is first loaded and preprocessed to produce a validated `processed_eeg` artifact. Feature extraction and PSD analysis both use this artifact and can execute in either order after preprocessing. The dependency structure is

$$\begin{aligned} \texttt{EEGFileLoader} &\rightarrow \texttt{EEGPreprocessor} \\ &\rightarrow \{\texttt{EEGFeatureAnalysis}, \texttt{EEGPSDPloter}\}. \end{aligned} \tag{25}$$

Equation 25 represents the required artifact dependencies without imposing an unnecessary order between the two independent downstream operations. The same analytical goals, artifact dependencies, and tool registry are used for both datasets.

F.2 Cross-Dataset Execution Results

Table 4 summarizes the six ISRUC-Sleep recordings. All recordings contain 19 channels sampled at 200 Hz, with durations ranging from 10,724 to 28,890 s.

Table 4: Real-data execution results on six ISRUC-Sleep recordings.

Subject	Status	Channels	Fs (Hz)	Duration (s)
sub-I001	Success	19	200	16,838
sub-I002	Success	19	200	10,724
sub-I003	Success	19	200	28,290
sub-I004	Success	19	200	28,890
sub-I005	Success	19	200	26,250
sub-I006	Success	19	200	26,910

Across the six ISRUC-Sleep recordings, BrainAgent-CGP processes 137,902 s (38.31 h) of EEG/PSG data. All six workflows complete successfully, resulting in 24 successful tool executions and no execution failures. Figure 2 shows PSD outputs generated from the validated `processed_eeg` artifacts, confirming successful downstream signal analysis.

Table 5 reports the corresponding results for Sleep-EDF Expanded. These recordings contain seven recorded signals sampled at 100 Hz, providing a different recording configuration from ISRUC-Sleep.

Across the six Sleep-EDF recordings, BrainAgent-CGP processes 500,760 s (139.10 h) of PSG data. All six workflows complete successfully, resulting in 24 successful tool executions and no execution failures.

Table 6 summarizes the combined results.

Across both datasets, BrainAgent-CGP processes 638,662 s (177.41 h) of real EEG/PSG data. All 12 workflows and all 48 tool executions complete successfully with no execution failures. The same

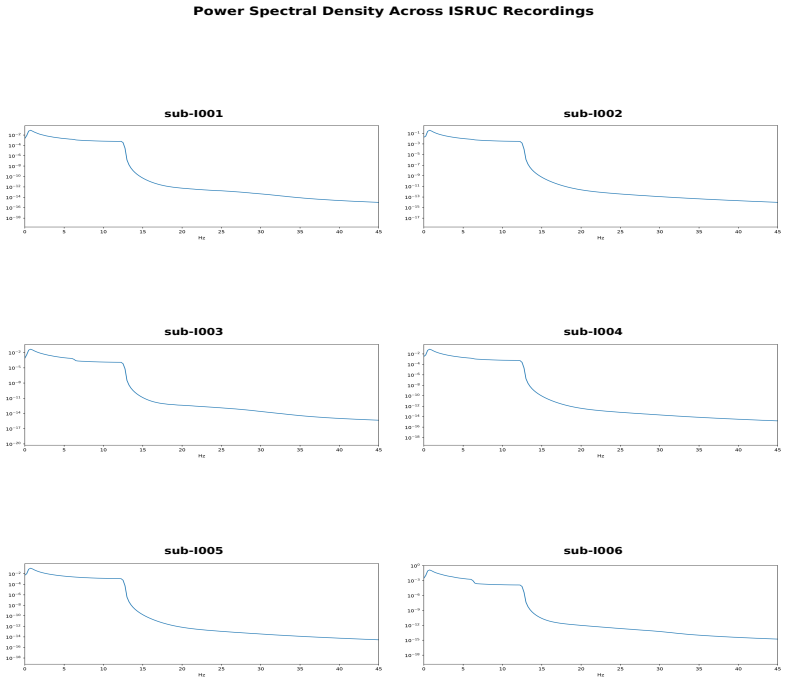


Figure 2: Power spectral density outputs generated by BrainAgent-CGP from the validated `processed_eeg` artifacts of six ISRUC-Sleep recordings.

Table 5: BrainAgent-CGP execution on six Sleep-EDF Expanded recordings.

Recording	Status	Channels	Fs (Hz)	Duration (s)
SC4001E0	Success	7	100	79,500
SC4002E0	Success	7	100	84,900
SC4011E0	Success	7	100	84,060
SC4012E0	Success	7	100	85,500
SC4021E0	Success	7	100	84,120
SC4022E0	Success	7	100	82,680

Table 6: Cross-dataset real-data execution results.

Dataset	Rec.	Channels	Fs (Hz)	Duration (h)	Tool Exec.	Failures
ISRUC-Sleep	6	19	200	38.31	24/24	0
Sleep-EDF	6	7	100	139.10	24/24	0
Overall	12	–	–	177.41	48/48	0

analytical goals, artifact dependencies, and tool registry are used for both datasets. These results provide initial evidence that the tested artifact-state workflow can operate across different recording configurations.

G Detailed Semantic-Robustness Results

G.1 Robustness to Paraphrased and Unseen Natural-Language Requests

We evaluate whether the semantic interpretation layer can recover the intended analytical goals when users express the same tasks using different natural-language formulations. The original 30-task benchmark is expanded with three variants per task, producing 90 additional requests. P1 uses a light paraphrase, P2 uses a conversational formulation, and U1 uses unseen wording that avoids the literal phrases in the controlled goal vocabulary. The analytical intent and reference workflow remain unchanged.

We first evaluate the deterministic controlled parser without modifying its phrase dictionary. As shown in Table 7, it achieves goal exact-match rates of 0.900 and 0.867 for P1 and P2, respectively, but fails on all 30 U1 requests. This result shows that literal phrase matching is not sufficient for lexically unseen formulations.

We then evaluate a hybrid semantic interpreter. The language model is restricted to mapping each request to the existing analytical-goal vocabulary; tool selection, agent assignment, dependency recovery, and workflow ordering remain the responsibility of CGP. The hybrid interpreter successfully parses all 90 requests and achieves an overall goal exact-match rate of 0.989, goal recall of 0.996, and goal precision of 1.000. Planning Success and Plan Validity Rate remain 1.00, routing accuracy remains 1.00, and the Invalid Tool Rate remains 0.00.

Only one request, L3_002_U1, produces an incomplete semantic mapping. The interpreter identifies feature extraction and sleep-report generation but omits the `visualize_psd` goal. The resulting workflow remains dependency-valid, and the overall Required-Tool Recall remains 0.998.

These results support the separation between semantic interpretation and executable planning. The semantic front end can be improved without changing the artifact-state planner, while CGP continues to recover dependencies and construct valid workflows.

Table 7: Semantic robustness on 90 paraphrased and unseen natural-language requests, with 30 requests in each category. EM: exact match; GR: goal recall; PS: planning success; PVR: plan validity rate; RTR: required-tool recall; RA: routing accuracy; ITR: invalid tool rate.

Interpreter	Variant	EM	GR	PS	PVR	RTR	RA	ITR
Controlled parser	P1	0.900	0.922	0.933	0.933	0.925	0.933	0.000
	P2	0.867	0.911	0.933	0.933	0.927	0.900	0.000
	U1	0.000	0.000	0.000	0.000	0.000	0.000	0.000
Hybrid semantic interpreter	P1	1.000	1.000	1.000	1.000	1.000	1.000	0.000
	P2	1.000	1.000	1.000	1.000	1.000	1.000	0.000
	U1	0.967	0.989	1.000	1.000	0.994	1.000	0.000

H Detailed Failure-Recovery Results

H.1 Failure-Aware Replanning

We evaluate whether BrainAgent-CGP preserves validated state and responds safely when tool execution fails. Because the production registry contains a single producer for each artifact, we use two settings. In three native cases, a required tool fails and no alternative producer is available. In three controlled recovery cases, an experiment-only fallback producer is introduced for the failed artifact. These fallback tools are used only for fault-injection evaluation and are not part of the production 21-tool registry.

Table 8 summarizes the results. In all three native cases, the failed tool is identified, previously validated artifacts are preserved, and the framework safely defers because no feasible workflow remains. In all three controlled recovery cases, BrainAgent-CGP blacklists the failed tool, replans

from the preserved artifact state, selects the fallback producer, and completes the requested goal. The Safe Deferral Rate and Recovery Rate are both 1.00.

Table 8: Failure-aware replanning under controlled tool failures. Native cases use the production registry, while controlled-recovery cases include an experiment-only fallback producer.

Setting	Cases	Goal Completion	Safe Deferral	Artifacts Preserved	Failed Tool Identified	Repeated Successful Tools
Native, no alternative	3	0/3	3/3	3/3	3/3	0
Controlled fallback	3	3/3	–	3/3	3/3	0
Overall	6	3/6	3/3 applicable	6/6	6/6	0

No previously successful tool is re-executed in any case. The Artifact Preservation Rate and Failed-Tool Identification Rate are both 1.00, while the Repeated Successful Tool Rate is 0.00. Mean fault-handling latency is 1.367 s. These results show that BrainAgent-CGP preserves valid state, recovers when an alternative path is available, and safely defers when no valid alternative remains.

I Additional Ablation Results

The effect of removing artifact constraints becomes stronger as task complexity increases. As shown in Table 9, PVR decreases from 0.50 for low-complexity tasks to 0.20 for medium-complexity tasks and 0.00 for high-complexity tasks. Planning Success also decreases to 0.30 for the high-complexity subset.

Table 9: Effect of removing artifact constraints across task complexity.

Complexity	PS	PVR	RTR	RA	ITR
Low	1.00	0.50	0.85	1.00	0.367
Medium	1.00	0.20	1.00	1.00	0.333
High	0.30	0.00	0.30	0.30	0.120

J Additional Experimental Results

J.1 Effect of Task Complexity

We examine how increasing workflow complexity affects BrainAgent-CGP. The benchmark contains 10 tasks at each complexity level. Level 1 includes short workflows, Level 2 contains sequential multi-step requests with implicit dependencies, and Level 3 contains longer multi-goal workflows that may span multiple agents.

Table 10: BrainAgent-CGP performance across task-complexity levels.

Level	Tasks	PS	PVR	RTR	Route	ITR	Latency (s)	Tools
L1	10	1.00	1.00	1.00	1.00	0.00	0.0004	2.40
L2	10	1.00	1.00	1.00	1.00	0.00	0.0029	3.60
L3	10	1.00	1.00	1.00	1.00	0.00	0.1110	6.40
Overall	30	1.00	1.00	1.00	1.00	0.00	0.0381	4.13

PS, PVR, RTR, and routing accuracy remain 1.00 across all complexity levels, while ITR remains 0.00. The effect of increasing complexity is mainly visible in workflow size and planning latency. Mean workflow length increases from 2.40 tools at Level 1 to 6.40 at Level 3, while mean planning latency increases from 0.0004 s to 0.1110 s. These results show that BrainAgent-CGP maintains dependency-valid planning as workflow complexity increases.

J.2 Workflow Efficiency

We further compare workflow size, redundant tool selections, and registered tool cost using the repeated 30-task planner experiment. Table 11 summarizes the results.

BrainAgent-CGP produces the most compact workflows among the evaluated methods, selecting an average of 4.13 tools with no redundant selections. The LLM-based planners select more tools on

Table 11: Workflow efficiency on the repeated 30-task benchmark.

Method	Tools ↓	Redundant ↓	TTC ↓
Single LLM	6.00	1.95	7.79
Hierarchical LLM	9.04	4.93	11.19
ReAct-style	4.75	1.05	6.12
BrainAgent-CGP	4.13	0.00	5.46

average and introduce redundant operations. BrainAgent-CGP also has the lowest mean registered tool cost of 5.46.

Figure 3 compares mean workflow length and redundant tool selections across the four planners.

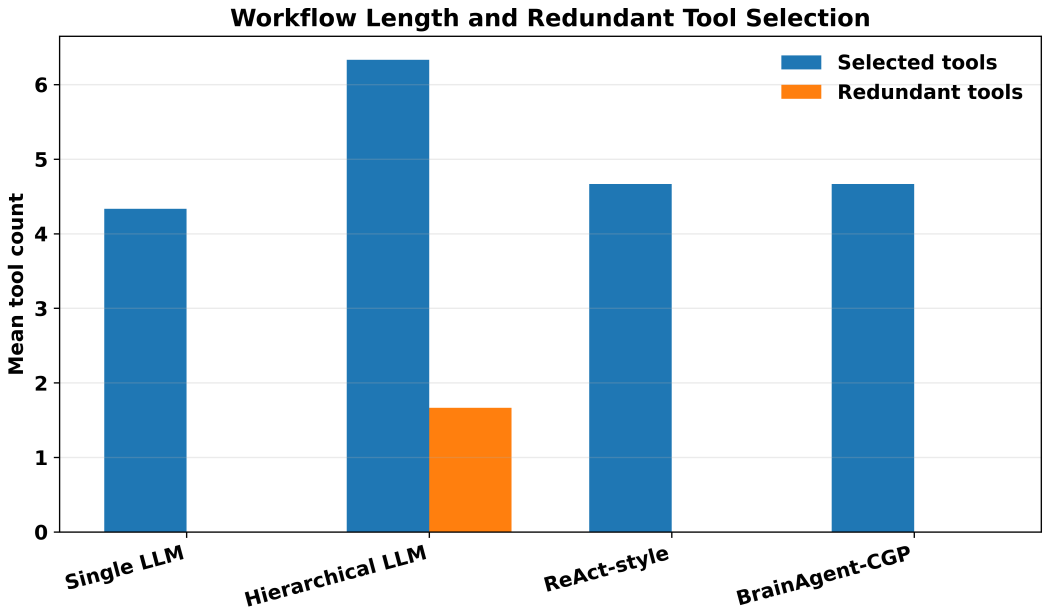


Figure 3: Mean workflow length and redundant tool selections on the repeated 30-task benchmark.

These results show that BrainAgent-CGP maintains complete required-tool coverage while avoiding unnecessary tool selections.