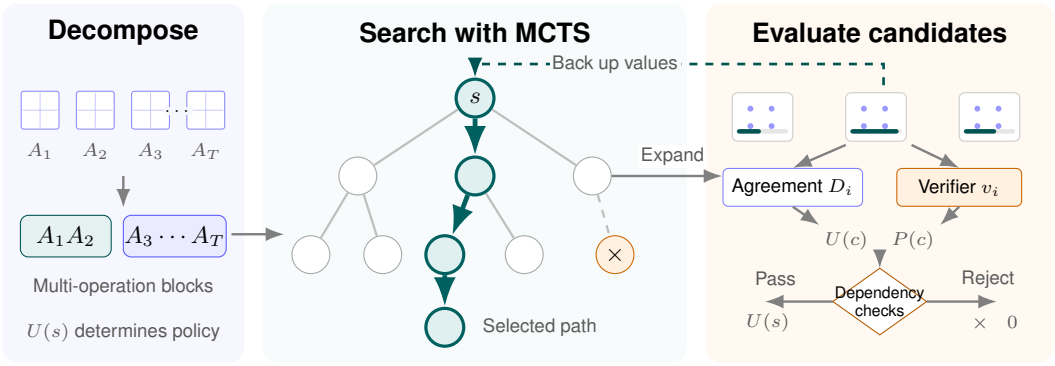
CONFIDENCE COORDINATION: ADAPTIVE DECOMPOSITION AND EXECUTION OF LONG-HORIZON TASKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models have made substantial progress in reasoning and task execution, but extending their capabilities to long-horizon operations requires careful problem decomposition and error control. Decomposition makes individual subproblems easier to solve but introduces additional coordination and state-transfer costs. Determining an appropriate decomposition therefore remains central to reliable long-horizon execution. This paper proposes the Confidence Coordination (CONCORD) framework, which addresses the problem through adaptive decomposition and Monte Carlo Tree Search-based execution. The framework explores alternative partial solutions using confidence estimates and dependency checks, allowing subproblems to contain multiple execution steps. The theoretical analysis provides a foundation for solvability and scalability by linking subproblem size and execution reliability to final-answer accuracy and cost over longer task horizons. To validate this theoretical advantage, experiments were conducted to compare CONCORD to state-of-the-art monolithic language models and prior decomposition approaches in a long-horizon challenge domain: matrix multiplication with varying numbers of matrices (horizon) and dimensions (complexity). CONCORD successfully solved the task at twice the horizon of the best-performing monolithic model with the same budget. More generally, the decomposition and execution configuration could be tuned to achieve lower cost and/or higher accuracy than alternative approaches. CONCORD thus provides a way to use given resources to solve more complex long-horizon problems.



(a) Adaptive decomposition (b) Search over partial solutions (c) Candidate evaluation

Figure 1: CONCORD overview. **(a)** The decomposition policy groups operations into subproblems. **(b)** MCTS explores alternative partial solutions within resource limits. **(c)** Agreement, verifier scores, and dependency checks guide candidate evaluation. Bars show schematic confidence scores. Together, decomposition and search organize dependent operations to balance reliability and cost.

1 INTRODUCTION

Recent progress in language-model reasoning has highlighted the importance of the reliable long-horizon execution. Shojaee et al. (2025) examined the limits of reasoning models and reported accuracy collapse as required number of task steps (e.g., in Towers of Hanoi) increased beyond

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.

model-dependent thresholds. With the goal of isolating the reliability of execution from planning and knowledge management, Sinha et al. (2026) found that small absolute error rates in per-turn accuracy led to catastrophic collaspse in long-horizon tasks, even with the most capable base models. MAKER (Meyerson et al., 2025) demonstrated how decomposition and error control could overcome these pitfalls by completing a Towers of Hanoi task with more than one million steps using focused agents and local error correction through per-step voting. Together, the studies motivate a closer examination of how the organization of model calls affects the length and complexity of tasks a system can complete.

Choosing subproblem size requires balancing reliability and cost. Smaller subproblems can make each model call easier, but they require more model calls and transfers of intermediate results (Meyerson et al., 2025; Liu et al., 2025). The additional model calls and transfers increase coordination cost. Decomposition must therefore be evaluated by both final-answer accuracy and cost (Kapoor et al., 2024; Wang et al., 2024). The aim is to choose a subproblem size that satisfies both the final-answer target accuracy and the available budget. However selection of subproblem size alone leaves two execution challenges. Candidate partial solutions must be assessed, but additional assessment consumes the same limited budget (Snell et al., 2024; Wang et al., 2024). Each selected partial solution must also provide compatible inputs and intermediate results for later subproblems. The challenge is to coordinate subproblem size and execution effort while preserving dependencies.

The Confidence Coordination (CONCORD) framework addresses these challenges through adaptive decomposition and confidence-guided Monte Carlo Tree Search (MCTS). The adaptive decomposition module splits subproblems until the policy chooses direct execution or reaches a resource limit. The MCTS controller uses confidence estimates to guide exploration among candidate partial solutions to each subproblem within resource limits. Dependency checks assess whether the candidate partial solutions use compatible inputs and intermediate results.

Figure 1 illustrates the interaction between decomposition, search, and evaluation. The decomposition policy groups operations into subproblems, and the controller explores candidate partial solutions under limits on rollouts, search depth, and model calls. Response agreement and verifier scores supply confidence estimates for search, while dependency checks reject structurally incompatible partial solutions. Along a completed path, each accepted block passes its resulting matrix to the next block. Horizon T and dimension d describe the task and determine limits on rollouts; search depth and model calls constrain execution effort.

The paper makes the following contributions:

- A method for organizing dependent long-horizon execution under resource limits: CONCORD integrates variable-size subproblems, confidence-guided MCTS, and dependency checks in a general search controller.
- A theoretical foundation for solvability and scalability: the analysis derives an accuracy bound and cost model linking subproblem size, execution reliability, and horizon.
- Empirical validation in a budget-constrained long-horizon challenge domain: CONCORD's recursive decomposition reached 2x longer horizons than monolithic models and prior decomposition methods.

Overall, the results suggest that adaptive decomposition, confidence-guided search, and careful use of verification can be combined to scale long-horizon execution of LLMs in a cost-effective way.

2 Background

Decomposition-based approaches to long-horizon execution must organize subproblems, assess candidate answers, and preserve the information needed by later subproblems. Prior work has addressed these requirements through fine decomposition, adaptive reasoning structures, and local error control. This section discusses prior work and identifies the remaining gaps that motivate CONCORD.

2.1 Decomposition and Adaptive Execution

MAKER combined maximal decomposition with local error correction (Meyerson et al., 2025). Each agent executed one predefined step. Repeated sampling of agents was run on each step until

an answer reached a specified voting margin. The system rejected responses with structural warning signs before voting. These simple techniques were sufficient to complete a Towers of Hanoi instance with more than one million dependent steps with zero errors. The main drawback of MAKER is that it requires a prespecified subtask decomposition, which can lead to innefficiencies both in cost and reliability. CONCORD addresses this limitation through an adaptive decomposition policy that permits multiple operations per subproblem.

Other prior work has investigated adaptive decomposition. Select-Then-Decompose, the closest related approach, selects a pre-defined strategy based on task characteristics and verifies the resulting answer (Liu et al., 2025). Graph of Thoughts represents intermediate answers and dependencies as a graph, supporting aggregation and refinement (Besta et al., 2024). ARIES uses policy agents to adapt graph transformations during execution (Gimenes et al., 2025). Pre-defined strategy selection, Select-Then-Decompose, and ARIES's graph adaptation leave adaptive decomposition unresolved for interdependent subproblems where precise subproblem-to-subproblem state transfer is critical. CONCORD addresses this gap by combining variable-size subproblems with explicit dependency checks, while using MCTS to boost reliability.

MCTS is a general tool for balancing exploration of candidate continuations against further evaluation of promising branches (Kocsis & Szepesvári, 2006). Reasoning via Planning uses a language model to simulate state transitions within MCTS (Hao et al., 2023). Language Agent Tree Search combines MCTS with reflection and environment feedback (Zhou et al., 2024). Applying such search to dependent execution also requires a policy defining the work assigned to each continuation. However, RAP and LATS do not provide an explicit policy for adapting subproblem size to balance execution reliability and coordination cost. CONCORD complements MCTS with an adaptive decomposition policy defining how much work each subproblem contains. MCTS uses confidence estimates to explore candidate partial solutions within resource limits, while dependency checks reject incompatible partial solutions.

2.2 Reliability and Resource Constraints

Confidence estimation of LLM outputs is an important area of research for LLM reliability and trustworthiness (Geng et al., 2024). For example, semantic density estimates response confidence from the distribution of sampled answers in semantic space (Qiu & Miikkulainen, 2024). However, although a confidence score can be useful for selecting the best among a set of answers, it is not on its own sharp enough to establish whether a subproblem answer supplies compatible inputs for the next subproblem. To sharpen reliability, CONCORD exploits three types of subproblem answer evaluation: response agreement (confidence estimation), verifier scores, and dependency checks.

Resource allocation must account for both answer assessment and decomposition (Liu et al., 2025). Research on inference-time computation found benefits from adapting effort to problem difficulty (Snell et al., 2024). Budget-aware evaluations showed the importance of comparing reasoning strategies at equivalent resource levels (Wang et al., 2024). For long-horizon execution, allocating more effort to an answer also leaves less budget for later subproblems. An open question is how to subproblem size and execution effort can be adapted to maintain acceptable accuracy over the full horizon. CONCORD enables theoretical analysis showing how a conditional accuracy bound can be maintained under a cost model capturing execution, coordination, and additional search (MCTS). The controller enforces the budget through limits on MCTS rollouts and model calls. These mechanisms support the theoretical and empirical analysis of configurations under resource constraints.

3 Method

CONCORD approaches the reliability and cost trade-off in long-horizon tasks by combining a task-specific decomposition policy with MCTS over partial solutions. The decomposition policy determines how much work a subproblem contains, and the search controller allocates exploration among the candidate continuations in each subproblem. The following subsections describe the decomposition and execution state, confidence estimation of candidate answers, and tree search for further exploration among candidate answers while keeping execution under resource limits.

3.1 Decomposition and execution state

Consider a finite task with initial state x_0 and T ordered steps. Step t has a deterministic target transition f_t and task-defined complexity $\chi_t \geq 0$:

$$x_t = f_t(x_{t-1}), \qquad t = 1, \ldots, T, \qquad y^\star = h(x_T), \tag{1}$$

where h extracts the output. Following MAKER's sequence-of-steps view (Meyerson et al., 2025), CONCORD permits variable block sizes and complexity profiles. The transferable state contains all information later steps require, including history when necessary. The task interface supplies block inputs and preserves target transitions under composition. A task-family parameter d gives a known step-complexity bound $\chi_t \leq \overline{\chi}(d)$, separately from horizon T. The interface defines complexity and d; neither specifies inference cost or LLM reliability.

A block $[i, j]$ applies $F_{i:j} = f_j \circ \cdots \circ f_i$ with size $g = j - i + 1$ and profile $(\chi_i, \ldots, \chi_j)$. Equal-size blocks can require different work. The recursive policy sends an unresolved subproblem and available predecessor answers to a splitter. The splitter keeps the subproblem or returns smaller ordered subproblems with dependency references. The interface defines their ranges and complete coverage. Recursion stops at direct execution, a depth or decomposition-call limit, or an optional short-question heuristic. The resulting leaves can contain several steps; granularity counts steps per executor call. The interface constructs leaf inputs from original inputs and predecessor answers. An LLM samples answers, and a task-specific rule composes child answers into parent candidates. A dependency graph records answer transfers. The explicit-graph policy retains supplied groupings and selects the first unresolved node with available predecessors in stable topological order.

A completed execution induces boundaries $0 = t_0 < \cdots < t_m = T$ and accepted blocks of size $g_\ell = t_\ell - t_{\ell-1}$. Execution with selected model answers, denoted by $\widehat{F}$, gives

$$\widehat{S}_0 = x_0, \qquad \widehat{S}_\ell = \widehat{F}_{t_{\ell-1}+1:t_\ell}(\widehat{S}_{\ell-1}), \qquad \sum_{\ell=1}^{m} g_\ell = T. \tag{2}$$

Complete ordered coverage, unchanged state transfer, and correct block answers imply $\widehat{S}_\ell = x_{t_\ell}$ and $h(\widehat{S}_m) = y^\star$ by induction (Appendix C). The argument assumes correct execution rather than guaranteeing LLM answers. Decomposition and MCTS represent separate choices: a search state s stores resolved subproblems and selected answers. An expansion can split, execute, and compose a whole block before exposing candidate continuations.

3.2 Confidence from candidate responses

For subproblem q, the executor samples $K \geq 1$ responses $r_1, \ldots, r_K$ and clusters equivalent answers into meaning classes $\mathcal{C}$ (Kuhn et al., 2023). Embedding similarity selects pairs for agreement checks, reducing provider requests. Each class retains the response with greatest mean agreement with its members. Classes merge paraphrases while preserving alternative continuations. Let response weights $w_i \geq 0$ satisfy $\sum_i w_i = 1$. Let $k_{ij} \in [0, 1]$ measure response agreement and $v_i \in [0, 1]$ denote verifier assessment of r_i for q. Inspired by semantic density (Qiu & Miikkulainen, 2024), CONCORD computes:

$$D_i = \sum_{j=1}^{K} w_j k_{ij}, \qquad U(c) = \frac{1}{|c|} \sum_{i \in c} \big[\alpha D_i + (1 - \alpha) v_i\big], \qquad P(c) = \left(\sum_{i \in c} w_i\right) U(c), \tag{3}$$

where $c \in \mathcal{C}$ and $\alpha \in [0, 1]$. The quality score $U(c)$ mixes batch support D_i with verification because agreement can reinforce shared errors. The prior $P(c)$ weights quality by class mass and considered as a heuristic for further decomposition.

CONCORD normalizes length-adjusted likelihoods when token probabilities exist (Qiu & Miikkulainen, 2024). Otherwise, it distributes smoothed class frequencies uniformly within each class (Appendix B). Length adjustment reduces penalties for longer responses; smoothing moderates small-batch frequency differences. The agreement matrix has unit diagonal and zero for unqueried pairs, yielding a partial estimate. With $K = 1$, $D_1 = 1$ supplies no cross-response evidence. Verification and dependency checks provide separate assessments.

3.3 TREE SEARCH AND DEPENDENCY CHECKS

Each instantiated class creates a child with an extended answer history and scores U and P. Each node stores a visit count n and value Q updated from explored descendants. For child c of s, CONCORD uses PUCT-style selection (Silver et al., 2017):

$$Q(c) + c_{\text{exp}} P(c) \frac{\sqrt{\max\{1, \sum_{u \in \text{ch}(s)} n(u)\}}}{1 + n(c)}, \tag{4}$$

where $\text{ch}(s)$ contains instantiated children and $c_{\text{exp}} > 0$ controls exploration. The prior guides early exploration, while visits reduce a child's bonus relative to its siblings. CONCORD uses unnormalized priors with $\sum_c P(c) \leq 1$, so total prior mass affects exploration strength. The adaptation supplies no new exploration principle or convergence guarantee. Progressive widening permits $\lfloor \kappa \max\{1, n(s)\}^{\beta} \rfloor$ children, with $\kappa \geq 1$ and $0 < \beta < 1$. The controller instantiates the highest-prior available class when permitted; otherwise, it follows Equation 4. Each rollout is one search iteration and adds at most one outer node. Candidate batches remain cached, and recursive expansion can require several model calls.

After expansion, the controller checks predecessor references and transferred answers against task-defined constraints as inputs become available. Let $\sigma(s) \in [0, 1]$ denote the largest recorded violation severity. The controller terminates branches with positive severity and assigns value zero. Otherwise, the controller assigns local value $U(s)$. Backup increments ancestor visits and takes the maximum child value among children with at least $n_{\min}$ visits. An ancestor retains its local score when no child qualifies. The threshold delays propagation from new children, but cached scores permit repeated visits without independent verification or removal of maximization bias (Appendix B).

The rollout limit N and outer depth limit bound search effort. Model-call accounting includes decomposition, execution, verification, and composition. The controller checks the call limit after each rollout, allowing batch overruns; a score-improvement heuristic can also stop search. The controller returns the highest-valued reached terminal passing dependency checks, favoring shallower ties. It reports rejection when all reached terminals fail checks and a partial result when none exists. Depth-limited terminals can remain incomplete: success requires complete coverage and a valid final answer. Algorithm 1 summarizes the controller.

4 THEORETICAL ANALYSIS: SOLVABILITY AND SCALABILITY

The analysis gives sufficient conditions for meeting an accuracy target and a monetary budget on ordered tasks with horizon T and complexity parameter d. The conditions concern the complete procedure, including decomposition and answer selection. The following subsections relate selected-block reliability to final-answer accuracy, determine the error allowance at a chosen granularity, and account for the cost of execution and search.

4.1 FROM BLOCK RELIABILITY TO FINAL-ANSWER ACCURACY

Fix a distribution over tasks with horizon T and complexity parameter d. Let $y^\star = h(x_T)$ be the target output, and let $\widehat{y}$ be CONCORD's returned output. An attempt with no valid output counts as a failure. Let E_0 denote completion with a valid ordered partition and correct input and state transfer. Assume $\Pr(E_0) \geq 1 - \eta$. The parameter $\eta \in [0, 1]$ bounds failures of completion or structural validity, including omitted steps and incorrect predecessor references. It is neither a decomposition accuracy score nor an answer-error probability. A procedure which guarantees completion and valid structure can set $\eta = 0$; otherwise, the analysis requires a justified bound for the chosen configuration.

Suppose every completed execution contains at most m accepted blocks. Let $\epsilon \in [0, 1]$ bound each selected block's error probability, conditional on E_0 and correctness of all preceding accepted blocks. The bound must cover the histories and block sizes produced by the complete selection procedure. For an execution with fewer than m blocks, append error-free empty blocks for the analysis.

Proposition 1 (Conditional final-answer accuracy)**.** *Under the completion, structural-validity, and conditional block-error assumptions,*

$$\Pr(\widehat{y} = y^\star) \geq (1-\eta)(1-\epsilon)^m. \tag{5}$$

The chain rule bounds the probability of valid completion and correct execution of every accepted block. Correct block composition then yields the target output. The conditional assumptions permit dependent block errors. Successful outputs after intermediate errors can only increase final-answer accuracy above the bound.

Equation 5 applies the standard multiplication of conditional success probabilities. Sinha et al. analyzed the expression p^T under constant independent step accuracy and no self-correction (Sinha et al., 2026). MAKER also related subtask success probabilities to full-task reliability before introducing voting (Meyerson et al., 2025). Setting $\eta = 0$ and $p = 1-\epsilon$ gives the familiar p^m expression as a lower bound. The analysis uses the standard argument to connect CONCORD's selected-block errors, structural failures, and costs.

4.2 Granularity and the Required Block Reliability

Let $1-\delta$ be the target final-answer accuracy, with $0 \leq \eta < \delta < 1$. Consider blocks of size g, except for a possibly shorter final block, so $m = \lceil T/g \rceil$. Let $\epsilon_T(g, d, K, N)$ bound the selected-block conditional error for the specified task distribution and procedure. Here K is the candidate sample count and N is the rollout limit. The error bound must cover the shorter final block and all relevant selection histories. Solving Equation 5 for the permissible error gives

$$\epsilon_T(g, d, K, N) \leq \epsilon_\star(m, \delta, \eta) := 1 - \left(\frac{1-\delta}{1-\eta}\right)^{1/m}, \qquad m = \lceil T/g \rceil. \tag{6}$$

The exact condition gives a per-block error allowance for the chosen horizon and accuracy target. Adaptive partitions also satisfy Equation 5 when m bounds their block count and ϵ bounds all their selected-block errors.

Larger blocks reduce the number of required answers but can increase selected-block error. More samples or rollouts help the sufficient guarantee only when they reduce the relevant error bounds. Both ϵ and η can depend on the horizon and the complete configuration. For fixed δ and η, the allowance $\epsilon_\star$ decreases on the order of $1/m$ as m grows. Reliable scaling therefore requires a procedure which reduces selected-block error as the number of blocks increases.

4.3 Monetary Cost and the Resource Budget

Define cost from the charges incurred during an attempt. Let J be its number of model calls, and let L_{jb} count tokens in billing category b for call j. Let π_{jb} be the applicable price per token. The disjoint categories distinguish ordinary input, output, and any separately priced cache usage. Let C_0 be setup cost and C_{other} include other charged resources, excluding setup and token charges. Total monetary cost is

$$C_{\text{total}} = C_0 + \sum_{j=1}^{J} \sum_{b \in \mathcal{B}} \pi_{jb} L_{jb} + C_{\text{other}}, \qquad \mathbb{E}[C_{\text{total}}] \leq B, \tag{7}$$

where $\mathcal{B}$ is the set of billing categories and B is the expected-cost budget. The expectation covers all attempts, including incomplete and incorrect outcomes. Call-specific token counts and rates account directly for differences in response length, model choice, and pricing.

A completed-path cost model illustrates the effect of granularity. Allocate each charge exactly once among setup, accepted-block work, and remaining search work. Let $c_{\text{coord}} \geq 0$ be coordination cost per accepted block and e_ℓ its execution and checking cost. Let $C_{\text{search}} \geq 0$ contain the remaining charges, including rejected alternatives. For a fixed model and sampling configuration, suppose $e_\ell = a(d)g_\ell + b(d)$, where $a(d) > 0$ and $b(d) \geq 0$. Then a completed execution with blocks of size g and a shorter final block when necessary has cost

$$\begin{aligned} C_{\text{total}} &= C_0 + \sum_{\ell=1}^{m} \left[c_{\text{coord}} + e_\ell\right] + C_{\text{search}} \\ &= C_0 + a(d)T + \left[c_{\text{coord}} + b(d)\right]\lceil T/g \rceil + C_{\text{search}}. \end{aligned} \tag{8}$$

The affine model assumes token charges grow linearly with block length at fixed d and fixed execution settings. Repeated full-history prompts or changing sampling effort require a different cost model. Larger blocks reduce per-block overhead while retaining the cost of processing all T steps.

A configuration meets the sufficient accuracy and expected-cost conditions when both Equations 6 and 7 hold. The joint conditions support comparisons of granularity and execution effort when justified error bounds and cost estimates are available. The conditional scaling construction in Appendix C states additional error-reduction and work assumptions explicitly.

5 Experiments

To empirically determine the effectiveness of CONCORD, experiments were run in a challenging long-horizon domain where even the most capable LLMs struggle. The experiments compared final accuracy and cost across horizons and dimensions. The following subsections define the task and evaluation metrics, describe the experimental setup, and comparisons of CONCORD to alternative approaches.

5.1 Task and evaluation metrics

An ideal benchmark for evaluating CONCORD would have the following properties: (1) the task horizon and per-step complexity should be controllable, to permit detailed behavioral analysis; (2) the space of possible decompositions should be enormous, so that an ideal decomposition cannot be known a priori; (3) state-of-the-art LLMs should fail miserably, establishing room for meaningful improvement; and (4) it should be inexpensive enough to run with an academic budget, to make future work accessible.

To that end, in this paper, matrix-chain multiplication is used as a controlled domain for studying dependent execution under resource limits. Earlier work on LLM execution used chains of 2×2 matrices to study sequential reasoning effort (Sinha et al., 2026). This paper extends the domain to include parameters for horizon T and dimension d. The task requires the final ordered product M_T from the initial state I_d. Increasing T adds dependent updates, while increasing d increases the arithmetic work (i.e., complexity) per update. Modular reduction keeps state entries bounded as the chain grows, so that the per-step complexity remains bounded. The task therefore isolates the impact decomposition and execution reliability under a bounded per-step complexity.

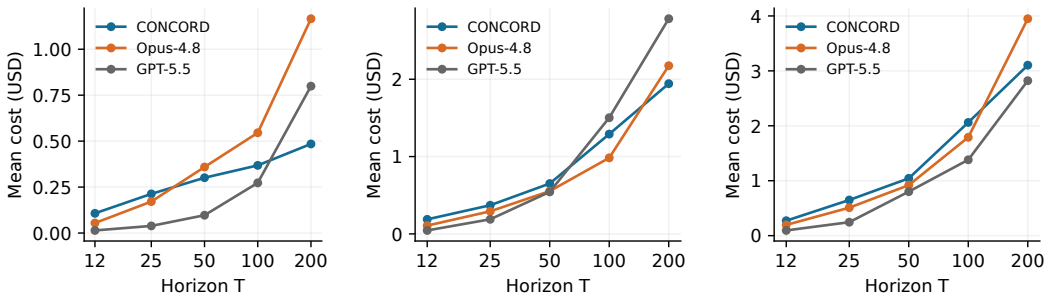
The evaluation measures exact final-matrix accuracy, mean recorded cost, and observed budget-constrained horizon. Final-matrix accuracy counts the fraction of observed outcomes whose returned matrix matches every entry of M_T. At $d = 1$, a zero factor makes every later exact product zero. Mean recorded cost measures USD per attempt, including incorrect returned answers. For system $\mathcal{A}$, let $\mathcal{T}_{\mathcal{A},d}$ contain the tested horizons at dimension d. Let $\widehat{p}_{\mathcal{A}}(d,T)$ denote observed final-matrix accuracy and $\widehat{C}_{\mathcal{A}}(d,T)$ denote mean recorded cost. The observed budget-constrained horizon summarizes qualifying tested settings. Comparisons must account for differences in tested grids and sample counts. For a mean-cost budget B and observed-accuracy threshold τ, the metric takes the largest qualifying horizon, or zero if none qualifies:

$$\widehat{H}_{\mathcal{A}}(B,d;\tau) = \max\left(\{0\} \cup \{T \in \mathcal{T}_{\mathcal{A},d} : \widehat{C}_{\mathcal{A}}(d,T) \le B,\ \widehat{p}_{\mathcal{A}}(d,T) \ge \tau\}\right). \tag{9}$$

5.2 Experimental setup

The experiments used arithmetic modulo 97 and dimensions $d \in \{1,2,3\}$. Comparisons were run for five horizons $T \in \{12,25,50,100,200\}$. Three independent trials were run for each (d,T) pair, for a total of 45 runs. To save compute, horizons shared prefixes of the same chain within each dimension. The recursive specialization used one MCTS rollout per subproblem (Appendix A). Its Opus-4.8 was used for execution, Sonnet-5 for decomposition, and Haiku-4.5 for verification.

The monolithic baselines used independent chains for the set budget: eight per (d,T) pair for Opus-4.8 at $d = 1,2$, six at $d = 3$, and five planned chains per dimension for GPT-5.5. Each model processed one matrix per turn and continued from its parsed state, while CONCORD received the full sequence. Recorded implementations of ARIES (Gimenes et al., 2025), Graph of Thoughts



(a) Dimension $d = 1$. (b) Dimension $d = 2$. (c) Dimension $d = 3$.

Figure 2: CONCORD used three repetitions of one chain, Opus used eight, eight, and six chains, and GPT used five planned chains, with four observed in the marked incomplete cells. Costs include incorrect returned answers and CONCORD runs carrying the truncation status.

(Besta et al., 2024), and Select-Then-Decompose (Liu et al., 2025) supplied ten-chain aggregates. Their horizons were set for the given budget as $\{100, 400, 1600\}$ at $d = 1$, $\{50, 100, 200\}$ at $d = 2$, and $\{10, 20, 50\}$ at $d = 3$. Recorded costs used the study's cache-aware price assumptions and an estimated GPT rate. Cells with incomplete monolithic traces used only traces reaching the target horizon and reported missing outcomes separately. The evaluation counted their returned matrices; full traces remain necessary to resolve the status.

5.3 Results

CONCORD returned $3/3$ correct final matrices at every scalar horizon and at $T = 12, 25$ in every dimension. CONCORD exceeded GPT's observed accuracy throughout $d = 2$ and after $T = 12$ at $d = 3$. Opus retained higher accuracy in several larger-dimension cells. CONCORD cost more than Opus at $T = 12$ in every dimension but less at $T = 200$ (Figure 2). At $T = 200$, CONCORD cost approximately \$0.48, \$1.94, and \$3.10 for $d = 1, 2, 3$, respectively. The amounts were 0.42, 0.89, and 0.79 times the corresponding Opus costs.

At $d = 1, T = 200$, CONCORD and Opus achieved perfect observed final-matrix accuracy. Mean cost divided by observed accuracy was approximately \$2.91 versus \$2.49 at $d = 2$, and \$9.31 versus \$4.74 at $d = 3$, for CONCORD and Opus respectively. Interpreting the ratios as expected retry costs requires stable success probabilities. Log-log cost fits over the five horizons gave CONCORD exponents of 0.51, 0.84, and 0.86, versus 1.04, 1.03, and 1.04 for Opus.

Observed horizon under a budget. On the five-point grid $T \in \{12, 25, 50, 100, 200\}$, CONCORD reached $T = 200$ for the scalar task with $3/3$ correct outcomes at \$0.485. Under the same \$0.50 mean-cost budget and 100% observed final-matrix accuracy requirement, GPT reached the longest qualifying monolithic horizon, $T = 100$, with $5/5$ correct outcomes at \$0.273. The twofold result applies to the five-point grid; scalar zero factors can erase intermediate errors. At $\tau = 0.5$ and $T \leq 200$, Table 1 reports dimension-dependent differences. At $d = 2$ under \$2, CONCORD reached $T = 200$, while Opus and Graph of Thoughts reached $T = 100$. At $d = 3$, the table reports CONCORD horizons of 50 and 100 under \$2 and \$4, respectively, compared with Opus horizons of 25 and 50. The decomposition baselines reached at most 20 and 50 under the corresponding budgets. The reporting cap retained only $T = 100$ for scalar decomposition baselines, although Graph of Thoughts and Select-Then-Decompose each returned $10/10$ correct outcomes at $T = 1600$ in the full records. CONCORD extended the observed budget-constrained horizon in selected pilot settings, and its lower per-attempt costs variably translate into lower costs per correct answer.

CONCORD treats the size of a subproblem as an execution decision, alongside the effort spent exploring candidate answers (Section 3). The analysis explains the resulting trade-off: larger blocks reduce coordination overhead but can require more effort to execute reliably (Section 4). Under stated assumptions, the accuracy and cost conditions identify configurations capable of meeting a final-answer accuracy target within an expected-cost budget. The practical implication is to choose decomposition and search effort together, using justified error bounds and cost estimates rather than

System	$2 per problem $d=1$	$d=2$	$d=3$	$4 per problem $d=1$	$d=2$	$d=3$
CONCORD	200	200	50	200	200	100
Opus-4.8	200	100	25	200	200	50
GPT-5.5	200	12	12	200	12	12
ARIES (Gimenes et al., 2025)	100	0	10	100	100	50
Graph of Thoughts (Besta et al., 2024)	100	100	20	100	100	20
Select-Then-Decompose (Liu et al., 2025)	100	0	20	100	0	20

Table 1: Observed budget-constrained horizon from Equation 9, with $\tau = 0.5$ and $T \leq 200$. CONCORD and monolithic entries use the five-point grid $\{12, 25, 50, 100, 200\}$; decomposition baselines use their recorded grids. Zero denotes an empty qualifying set.

subproblem size alone (Equations 6 and 7). Prior work on difficulty-dependent inference supports adaptive resource allocation, while CONCORD raises the further question of how to allocate effort across dependent subproblems (Snell et al., 2024).

The matrix-chain experiments provide preliminary evidence for the framework, with important limits (Section 5). The pilot repeated one chain per dimension, used one MCTS rollout per subproblem, and compared systems under same protocols. The measured results support conclusions about the tested configurations; broader applications remain proposals.

6 Discussion and Conclusion

CONCORD addresses a central challenge in long-horizon execution: organizing dependent work to meet accuracy and cost requirements (Meyerson et al., 2025; Kapoor et al., 2024). Select-Then-Decompose established the value of adapting decomposition to task characteristics and resource demands (Liu et al., 2025). CONCORD combines variable-size subproblems with confidence-guided MCTS and dependency checks. Its analysis connects block reliability and structural validity to final-answer accuracy under explicit assumptions. The resulting framework supports reasoning about the resources required for reliable execution. The matrix-chain pilot identified promising accuracy–cost trade-offs, although the one-rollout configuration and differing baseline protocols leave the benefits of repeated search unmeasured.

The broader practical promise of CONCORD is to make complex automated workflows more accessible under limited resources. For example, a research assistant could coordinate data collection, computational analysis, and report preparation through subproblems with explicit dependencies. Task-specific tools could execute calculations (Khot et al., 2023), while checks could assess intermediate outputs before later stages use them (Lightman et al., 2023). If application studies establish reliable execution at lower cost, small research teams and public-interest organizations could undertake more extensive analyses within existing budgets. Reducing manual coordination could also leave people more time for interpretation and decisions requiring human judgment.

Future work should develop practical applications through grounded execution and controlled evaluation. SayCan connected language planning to executable robot skills (Ahn et al., 2022), while ReAct combined reasoning with actions and external feedback (Yao et al., 2023). CONCORD could complement these approaches by grouping dependent operations and exploring alternative partial solutions within resource limits. Potential applications include coordinating robot skills and managing workflows involving information retrieval and computational tools. Each application requires explicit state transfer and task-specific checks. Evaluations across independent tasks should measure intermediate correctness and search benefits (Lightman et al., 2023; Kapoor et al., 2024), while calibrated error estimates should guide resource allocation (Snell et al., 2024).

CONCORD provides a framework for connecting decomposition and search decisions to the reliability and cost of long-horizon execution. Its broader contribution is a basis for designing execution procedures around explicit accuracy targets and resource constraints. Further validation can turn the conditional analysis and initial empirical evidence into practical guidance for longer, more varied tasks.

AI USE STATEMENT

In this work, we used generative AI tools to assist with drafting, grammar-checking, lexical consistency, and maintaining content coherence. We did not use generative AI tools to develop the theoretical and empirical analysis. Additionally, we used generative AI tools for manuscript organization, literature identification, figure and table generation, and writing analysis code. After completing author review, we carefully reviewed all AI-assisted work. We checked mathematical claims and proofs against their assumptions, verified reported results against experimental records, checked citations against original sources, and tested AI-assisted analysis code. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Code, evaluation harnesses, figure and analysis scripts, and the saved per-generation label records are available at https://anonymous.4open.science/r/ConCord-1E6B. The appendices give metric definitions, model and judge identifiers, training settings, prompt templates, data construction, and the analysis procedures.

REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of Thoughts: Solving Elaborate Problems with Large Language Models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690, March 2024. ISSN 2159-5399. doi: 10.1609/aaai.v38i16.29720. URL http://dx.doi.org/10.1609/aaai.v38i16.29720.

Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. A survey of confidence estimation and calibration in large language models. In *Proceedings of the 2024 conference of the north American chapter of the association for computational linguistics: human language technologies (volume 1: long papers)*, pp. 6577–6595, 2024.

Pedro Gimenes, Zeyu Cao, Jeffrey Wong, and Yiren Zhao. ARIES: Autonomous Reasoning with LLMs on Interactive Thought Graph Environments, 2025. URL https://arxiv.org/abs/2502.21208.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with Language Model is Planning with World Model, 2023. URL https://arxiv.org/abs/2305.14992.

Sayash Kapoor, Benedikt Stroebl, Zachary S. Siegel, Nitya Nadgir, and Arvind Narayanan. AI Agents That Matter, 2024. URL https://arxiv.org/abs/2407.01502.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. Decomposed Prompting: A Modular Approach for Solving Complex Tasks, 2023. URL https://arxiv.org/abs/2210.02406.

Levente Kocsis and Csaba Szepesvári. Bandit Based Monte-Carlo Planning. In Johannes Fürnkranz, Tobias Scheffer, and Myra Spiliopoulou (eds.), *Machine Learning: ECML 2006*, pp. 282–293, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-46056-5.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic Uncertainty: Linguistic Invariances for Uncertainty Estimation in Natural Language Generation, 2023. URL https://arxiv.org/abs/2302.09664.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's Verify Step by Step, 2023. URL https://arxiv.org/abs/2305.20050.

Shuodi Liu, Yingzhuo Liu, Zi Wang, Yusheng Wang, Huijia Wu, Liuyu Xiang, and Zhaofeng He. Select-Then-Decompose: From Empirical Analysis to Adaptive Selection Strategy for Task Decomposition in Large Language Models. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 5454–5477, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.278. URL https://aclanthology.org/2025.emnlp-main.278/.

Elliot Meyerson, Giuseppe Paolo, Roberto Dailey, Hormoz Shahrzad, Olivier Francon, Conor F. Hayes, Xin Qiu, Babak Hodjat, and Risto Miikkulainen. Solving a Million-Step LLM Task with Zero Errors, 2025. URL https://arxiv.org/abs/2511.09030.

Xin Qiu and Risto Miikkulainen. Semantic Density: Uncertainty Quantification for Large Language Models through Confidence Measurement in Semantic Space, 2024. URL https://arxiv.org/abs/2405.13845.

Parshin Shojaee, Iman Mirzadeh, Keivan Alizadeh vahid, Maxwell Horton, Samy Bengio, and Mehrdad Farajtabar. The Illusion of Thinking: Understanding the Strengths and Limitations of Reasoning Models via the Lens of Problem Complexity. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 108018–108059. Curran Associates, Inc., 2025. doi: 10.52202/085713-3603. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/9b26ad15462c81548c0689188d2e8018-Paper-Conference.pdf.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354–359, October 2017. ISSN 1476-4687. doi: 10.1038/nature24270. URL http://dx.doi.org/10.1038/nature24270.

Akshit Sinha, Arvindh Arun, Shashwat Goel, Steffen Staab, and Jonas Geiping. The Illusion of Diminishing Returns: Measuring Long Horizon Execution in LLMs, 2026. URL https://arxiv.org/abs/2509.09677v3.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM Test-Time Compute Optimally can be More Effective than Scaling Model Parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Junlin Wang, Siddhartha Jain, Dejiao Zhang, Baishakhi Ray, Varun Kumar, and Ben Athiwaratkun. Reasoning in Token Economies: Budget-Aware Evaluation of LLM Reasoning Strategies, 2024. URL https://arxiv.org/abs/2406.06461.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing Reasoning and Acting in Language Models, 2023. URL https://arxiv.org/abs/2210.03629.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language Agent Tree Search Unifies Reasoning, Acting, and Planning in Language Models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 62138–62160. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhou24r.html.

Algorithm 1 CONCORD search with a task-specific expansion policy

Require: Task, dependency graph, expansion policy, sample count K, rollout limit N, depth and call limits
Initialize root s_0 with empty answers; initialize terminal set $\mathcal{T}$
for $r = 1, \ldots, N$ **do**
  Descend from s_0 using Equation 4 until a terminal or expandable node s
  if s is expandable **then**
    Obtain or reuse a candidate batch from the expansion policy at s
    Group responses; compute $U(c)$ and $P(c)$ using Equation 3
    If widening permits, instantiate one available class and set s to the new child
  end if
  Apply task termination and depth rules to s
  Evaluate applicable dependencies to obtain $\sigma(s)$
  if $\sigma(s) > 0$ **then**
    Mark s terminal-rejected; set $V \leftarrow 0$
  else
    Set $V \leftarrow U(s)$
  end if
  Increment $n(s)$ and set $Q(s) \leftarrow V$; back up to the root
  Record s in $\mathcal{T}$ if terminal; update costs and stopping statistics
  if call limit exceeded or score-improvement stopping rule fires **then**
    Record stopping status and break
  end if
end for
Return the highest-valued terminal passing the dependency checks, or the recorded rejection/partial result

A MATRIX-CHAIN EXECUTION SPECIALIZATION

The matrix-chain experiments used an additional recursive pipeline within the CONCORD project. The specialization exposes the range-resolution and state-composition operators described in Section 3.1. The recursive pipeline supplies an expansion policy to the same MCTS controller. The following details identify the expansion procedure used in the measured runs.

A.1 CONFIGURATION AND EXECUTION ORDER

All 45 supplied CONCORD runs recorded `solver=pipeline` and one rollout. The final configuration set a maximum split depth of 3, at most 5 children per split, a per-node call ceiling of 600, and one executor sample per leaf. The short-question stopping heuristic was disabled during the study. The splitter emitted inclusive input ranges and dependency references. A source resolver supplied the original matrix values and specified multiplication with the incoming state on the left.

The pipeline recursively partitioned blocks and executed leaves in dependency order. A later leaf consumed the earlier leaf's returned state. Composite answers used the result from the child covering the largest input index. When the deterministic resolver returned an answer, the implementation bypassed the block-level language-model combiner and its verifier. Leaf verification remained active. When the resolver lacked the required child answer, the implementation could fall back to the language-model combination path. The deterministic result established the assembly rule; arithmetic correctness still depended on the executor and the incoming state.

The depth cap permitted leaves containing multiple factors. For example, the $d = 1, T = 200$ runs recorded 50 or 51 executor calls. Such counts are compatible with grouped factors, while complete leaf-span traces are needed to reconstruct the full distribution of leaf sizes. The measurements provide no controlled comparison between the resulting granularity and an optimal fixed granularity.

The configuration also contained general MCTS parameters, including a nominal budget of 64, an exploration coefficient of 3, and a minimum child visit count of 3. The solver field identifies the recursive expansion policy, and the one-rollout records limit the extent of tree exploration. The matrix-chain results therefore supply evidence for the recursive specialization within the controller. Experiments varying the general search controller are needed to measure its separate contribution.

B SEARCH IMPLEMENTATION AND RESOURCE ACCOUNTING

The implementation separates the search controller from its expansion policy. The following details specify response weighting, value propagation, and stopping behavior so the main algorithm can be related to the supplied code. The complexity bounds account for the implemented candidate comparisons and distinguish search-tree size from work inside a recursive expansion.

B.1 RESPONSE WEIGHTS AND CLASS REPRESENTATIVES

When token log probabilities are available, let $L_i \geq 1$ be the response length and ℓ_{it} its token log probability. The normalized response weight is

$$\widetilde{w}_i = \exp\left(\frac{1}{L_i}\sum_{t=1}^{L_i}\ell_{it}\right), \qquad w_i = \frac{\widetilde{w}_i}{\sum_{j=1}^{K}\widetilde{w}_j}. \tag{10}$$

The length normalization reduces the direct dependence of a sequence likelihood on its length. These weights summarize model likelihoods and require separate calibration before interpretation as correctness probabilities. If a response lacks token probabilities, the implementation uses a raw weight of one before normalization; a zero total weight triggers uniform weights.

When token probabilities are unavailable, let $J = |\mathcal{C}|$ be the number of nonempty meaning classes and let $a_{\mathrm{L}} \geq 0$ be a smoothing constant. A class c with $|c|$ responses receives mass $\mu(c)$, distributed equally among its members:

$$\mu(c) = \frac{|c| + a_{\mathrm{L}}}{K + a_{\mathrm{L}}J}, \qquad w_i = \frac{\mu(c)}{|c|} \quad (i \in c). \tag{11}$$

Since the classes partition the K responses, $\sum_c \mu(c) = 1$ and $\sum_i w_i = 1$. Equation 3 then gives $0 \leq D_i, U(c), P(c) \leq 1$ and $\sum_c P(c) \leq 1$. The implementation uses the unnormalized class priors in the selection score. Their total magnitude consequently affects the strength of exploration along with c_{exp}.

The clustering procedure first groups response embeddings, then confirms selected pairs using the configured agreement kernel. Each class is represented by the response with the greatest mean agreement with its class members. The stored kernel matrix has unit diagonal and zero entries for unqueried pairs. Computing density from the stored matrix therefore uses a partial agreement estimate. The available implementation supports configurable embedding and kernel providers and defaults to lexical providers when replacements are absent.

B.2 BACKUP, DEPTH, AND TERMINATION

After evaluating a visited node s, the controller increments $n(s)$ and sets its value to the evaluated score. It then visits ancestors in reverse path order. Define the eligible children of an ancestor u as $\mathcal{E}(u) = \{c \in \mathrm{ch}(u) : n(c) \geq n_{\min}\}$. The default backup is

$$Q(u) = \begin{cases} \max_{c\in\mathcal{E}(u)} Q(c), & \mathcal{E}(u) \neq \varnothing, \\ U(u), & \mathcal{E}(u) = \varnothing. \end{cases} \tag{12}$$

The root starts with local score zero. Repeated visits can reuse a cached evaluation, so the visit count measures search activity rather than independent verification trials. The guard delays a child's influence on its parent; controlling maximization bias requires additional assumptions on score errors.

Let G' be the condensed dependency graph, $L_{G'}$ its longest path length in edges, $s_D \geq 0$ a depth slack, and J_{call} a nominal call limit. The adaptive depth heuristic used by the implementation is

$$D_{\max} = \max\left\{1, \min\left\{L_{G'} + s_D, \max\left\{1, \left\lfloor \frac{J_{\mathrm{call}}}{K} \right\rfloor\right\}\right\}\right\}. \tag{13}$$

The current configuration reuses the rollout-limit value for J_{call}. The notation separates the two roles because a rollout can require several calls. The depth estimate is heuristic: graph critical-path length differs from the number of subproblems resolved along a sequential search path, and verification or

recursive expansion can cost more than K calls. A depth cap can therefore stop an incomplete path. Completion belongs to the structural event E_0 in the reliability analysis.

For online stopping, let $U_z^\star$ be the largest local score observed at search depth z. When scores are available at adjacent depths, the gain estimate is $\widehat{\Delta}_z = U_z^\star - U_{z-1}^\star$. In adaptive mode, the implementation considers stopping after depth $D_{\max}$ has been reached somewhere in the tree and at least $\max\{8, \lfloor N/4 \rfloor\}$ rollouts have completed. It stops when $\widehat{\Delta}_z < \lambda \max\{1, K\}$, with $\lambda \geq 0$ expressing the minimum score gain per estimated call. The estimate compares scores at different depths and serves as a heuristic. The measured call limit is checked after each rollout, so stopping can occur after a batch has crossed the limit.

The explicit-graph decomposer selects the first unresolved node with resolved predecessors in a stable topological order. Its granularity argument is currently an interface parameter and leaves the graph-defined grouping unchanged. The recursive expansion policy instead obtains a split tree from a language-model splitter and executes its leaves in dependency order. Consequently, task-dependent grouping comes from the expansion policy, and the current explicit-graph implementation supplies no evidence of confidence-driven optimization over numerical matrix block sizes. Algorithm 1 describes the shared controller; the recorded matrix runs exercised the recursive policy with one rollout.

B.3 Search work and memory

At most one child is instantiated in each rollout, so N rollouts create at most $N + 1$ search nodes including the root. Progressive widening gives at most $W_N = \lfloor \kappa \max\{1, N\}^\beta \rfloor$ children per node. Selection and backup scan child sets along paths of depth at most $D_{\max}$, giving $O(ND_{\max}W_N)$ controller work under constant-time node access. The bound concerns the outer search tree; a recursive expansion can generate a separate internal split tree.

For a flat expansion batch, K executor samples require generation, embedding, and verification. The supplied clustering implementation performs dense embedding comparisons and can query $O(K^2)$ agreement pairs in the worst case. If H bounds embedding dimension and $c_{\text{gen}}, c_{\text{emb}}, c_{\text{ver}}, c_{\text{pair}}$ bound work per generation, embedding, verification, and agreement query, respectively, a conservative per-batch bound is

$$W_{\text{batch}} = O\left(K(c_{\text{gen}} + c_{\text{emb}} + c_{\text{ver}}) + K^2(H + c_{\text{pair}})\right). \tag{14}$$

Input lengths, token limits, graph operations, and state copying must be included in the corresponding work terms. Each rollout invokes at most one node-expansion operation, which requests at most two flat candidate batches, including its refill path. Thus the flat-policy work is bounded by $O(ND_{\max}W_N + NW_{\text{batch}})$ plus graph construction, decomposition, gating, and state-management work. For the recursive policy, the actual internal calls replace the flat-batch term. Stored search nodes are $O(N)$, while response queues, kernel matrices, answer histories, caches, and internal split trees contribute additional memory.

C Proofs and interpretation of the analysis

The proofs connect exact block composition, accepted-answer errors, and total resource requirements. All probability claims concern the execution selected by the complete policy, including search and answer selection. The final scalability construction makes explicit the local error-reduction assumption required to maintain accuracy at increasing horizons.

C.1 Correctness of block composition

Consider an ordered partition $0 = t_0 < t_1 < \cdots < t_m = T$. Assume the resolver supplies exactly the factors in every block, the incoming state is transferred unchanged, and each accepted block correctly applies F. The base state is $\widehat{S}_0 = I_d = M_0$. If $\widehat{S}_{\ell-1} = M_{t_{\ell-1}}$, then

$$\widehat{S}_\ell = F_{t_{\ell-1}+1:t_\ell}(M_{t_{\ell-1}}) = M_{t_{\ell-1}} A_{t_{\ell-1}+1} \cdots A_{t_\ell} = M_{t_\ell} \pmod{97}. \tag{15}$$

Induction yields $\widehat{S}_m = M_T$. The same argument applies recursively to a block partitioned into contiguous children: the final child's state is the parent block's result. Deterministic composition preserves correct execution under the stated coverage and arithmetic assumptions.

C.2 Accepted-answer probability bound

Let C_ℓ denote correctness of accepted block ℓ, and let $C_{<\ell} = \bigcap_{j<\ell} C_j$. Assume $\Pr(E_0) \geq 1 - \eta$ and $\Pr(C_\ell \mid E_0, C_{<\ell}) \geq 1 - \epsilon_\ell$ whenever the conditioning event has positive probability. A bound holding uniformly over the histories induced by adaptive selection implies the stated conditional bound. When every prefix conditioning event has positive probability, the chain rule gives

$$\Pr\left(E_0 \cap \bigcap_{\ell=1}^{m} C_\ell\right) = \Pr(E_0) \prod_{\ell=1}^{m} \Pr(C_\ell \mid E_0, C_{<\ell}) \geq (1-\eta) \prod_{\ell=1}^{m} (1-\epsilon_\ell). \tag{16}$$

If a prefix event has probability zero, the inequality follows by considering the first zero-probability prefix; the corresponding lower bound is zero. Correct structure and correct blocks imply the correct final answer by Equation 15. The elementary inequality $\prod_\ell (1-\epsilon_\ell) \geq 1 - \sum_\ell \epsilon_\ell$, together with nonnegativity, establishes Proposition 1. For a random number of blocks bounded by m, append error-free empty blocks to the accepted path. Raw sample error rates can be substituted only after establishing their relationship to errors in the selected answers.

C.3 Cost, scaling, and the final partial block

Write $T = qg + r$ with integers $q \geq 0$ and $0 \leq r < g$. A fixed-granularity execution has q full blocks and one block of length r when $r > 0$. With $e(u, d) = a(d)u + b(d)$, summing actual block lengths gives $\sum_\ell e(g_\ell, d) = a(d)T + b(d)\lceil T/g \rceil$. Substitution into Equation 7 proves Equation 8. Charging every block as a full block instead overcounts the final block by $a(d)(g - r)$ when $r > 0$.

For fixed g, d and coefficients, the accepted-path cost is $\Theta(T)$. If a reference costs $a_0 T$ and a candidate model costs $C_0 + a_1 T$, their costs are equal at $T = C_0/(a_0 - a_1)$ when $a_0 > a_1$; the candidate is cheaper above the crossing. The comparison assumes the stated cost models and requires a separate accuracy comparison. For positive C_0, a, the local log-log slope is

$$\frac{\mathrm{d}\log(C_0 + aT)}{\mathrm{d}\log T} = \frac{aT}{C_0 + aT} < 1. \tag{17}$$

A slope below one over a finite range can therefore arise from amortizing setup cost even when total work grows linearly. Source indices also require $O(\log T)$ bits, and rereading a full source at multiple decomposition levels can add horizon-dependent overhead. Any asymptotic model must include those costs.

C.4 A conditional construction for reliable scaling

Fix block-size bound g and matrix dimension d, and put $m = \lceil T/g \rceil$. Suppose a complete acceptance procedure has error at most $A\exp(-\gamma K)$ per block, uniformly over relevant histories after selection, where $A, \gamma > 0$. Suppose also the structural and completion error remains at most $\eta < \delta$. Choosing

$$K \geq \max\left\{1, \left\lceil \frac{1}{\gamma} \log \frac{Am}{\delta - \eta} \right\rceil\right\} \tag{18}$$

ensures $mA\exp(-\gamma K) \leq \delta - \eta$, so Equation 6 gives final-answer accuracy at least $1 - \delta$. Since $m = \Theta(T)$ for fixed g, the required sample count is $O(\log(T+1))$. If total work attributable to each block is $O(K)$ and all remaining search work is $O(mK)$, the construction costs $O(T\log(T+1))$. With a dense $O(K^2)$ confidence computation per block, the corresponding computational bound becomes $O(T\log^2(T+1))$ under the same bounded-cost assumptions. Token charges and local computation should be accounted for in their respective resource units.

The exponential error law is an assumption about the complete acceptance procedure. Establishing the law requires a calibrated error model or an additional proof for a specified sampler and selector. CONCORD's current confidence scores, repeated visits, and supplied one-rollout matrix runs establish neither the exponential law nor a convergence rate for accepted-answer correctness. The construction identifies a concrete route to scalable execution and the evidence required to support it.