Resampled Thought Trees: Measuring Intermediate-State Utility in Tree-of-Thought Search

Nishka Parikh Faraaz Rahman Jithin Rajan Raunak Sood Weston Trautmann
University of Southern California
{nmparikh, faraazra, jrajan, raunakso, wtrautma}@usc.edu

Abstract

Tree-of-Thought (ToT) reasoning treats language-model inference as search over intermediate thoughts rather than as a single linear chain. Existing ToT work typically evaluates final task accuracy or search efficiency, implicitly relying on local branch scores or pruning heuristics to decide which thoughts are worth exploring. This leaves a central question underexplored: *which intermediate thoughts actually matter for eventual success?* We introduce **Resampled Thought Trees**, a diagnostic framework for estimating the downstream utility of intermediate ToT states. Given a logged search tree, we freeze an internal node, restart search from that node multiple times, and estimate the conditional probability of eventually reaching a correct solution. We then validate these node-utility estimates with branch ablation, asking whether removing a branch changes binary solve rate or reduces the number of successful solution leaves.

We evaluate this framework on two controlled ToT-style tasks with different search geometries. In Game24, candidate arithmetic operations are generated deterministically and ranked by a locally served LLaMA-family model. Across 100 puzzles, two temperature settings, 1,600 node summaries, and 16,000 resampling trials, we find a polarized utility landscape: many states are dead ends, while others are critical or redundant-viable. A 792-row branch-ablation study shows that dead-end branches can consume search budget, while critical and redundant-viable branches preserve solve rate or solution mass. We then extend the analysis to crossword-style fill search, using 20 expanded crossword puzzles, candidate-bank generation with crossing-letter constraints, and LLM ranking. Across 387 branch-diverse crossword node summaries and 3,870 resampling trials, we find a different geometry: wrong early fills often die immediately, while deeper surviving states become increasingly reliable as constraints accumulate and remaining search shrinks. Together, the results show that intermediate thoughts should be treated as measurable search states, not merely locally ranked text fragments. Node utility depends on task structure, depth, branch rank, search policy, and constraint propagation.

1 Introduction

Large language models (LLMs) can improve on complex reasoning tasks when prompted to generate intermediate reasoning steps, as in Chain-of-Thought (CoT) prompting [1]. However, CoT remains fundamentally linear: once a model commits to an early reasoning direction, later tokens unfold from that single trajectory. Tree-of-Thought (ToT) reasoning generalizes this approach by treating reasoning as search over intermediate states, allowing a model-driven controller to generate, score, and expand multiple candidate thoughts [2, 3]. This makes ToT attractive for planning, arithmetic search, puzzle solving, and other settings where alternatives matter.

The practical difficulty is branch selection. A ToT controller must decide which intermediate thoughts deserve further expansion. Existing approaches often rely on local model scores, local self-evaluation, or pruning criteria such as semantic similarity [5]. These methods can improve efficiency, but they leave a deeper question unanswered: *does a locally plausible thought actually preserve useful futures?* A branch can be ranked highly by the model while leading to a dead end; another branch can be locally under-ranked but preserve many successful completions. Final-answer accuracy alone cannot reveal these distinctions.

This paper studies ToT search trees as empirical objects. Our central question is:

> Given a Tree-of-Thought search tree, can we empirically identify which intermediate thoughts are useful, useless, redundant, mixed, or misleading?

We propose **node-level resampling** as a direct diagnostic. Instead of only observing whether a full search succeeds, we freeze a particular intermediate node, restart the search from that exact state multiple times, and estimate the probability that search succeeds from there. This transforms an intermediate thought from a local textual step into a measurable conditional object:

$$\hat{p}(v) \approx P(\text{success} \mid \text{state } v).$$

We then pair resampling with **branch ablation**, which asks whether blocking a selected branch changes the behavior of the whole tree. The two analyses complement each other: resampling estimates the utility of a state if reached, while ablation tests whether that branch contributes causally to the original search.

Our findings show that a useful analysis of ToT requires more than a slogan like "early nodes are good." Node utility is task-dependent. In Game24, a shallow arithmetic task with exactly three operation steps, early arithmetic choices often open or close large regions of future success. In crossword-style fill search, wrong early fills often collapse immediately under crossing constraints, while deeper surviving states become more reliable because invalid futures have already been pruned and fewer decisions remain. Thus, node utility depends on *task geometry*: the interaction between depth, local rank, search policy, constraint propagation, and solution redundancy.

Our contributions are:

1. We introduce a diagnostic framework for estimating intermediate-state utility in Tree-of-Thought search via node-level resampling.
2. We validate node-utility categories with a branch-ablation study that distinguishes binary solve-rate effects from loss of successful solution mass.
3. We provide a large controlled Game24 study with 1,600 node summaries, 16,000 resampling trials, and 792 ablation rows.
4. We extend the analysis to deeper crossword-style fill search, showing that utility curves differ substantially under constraint propagation.
5. We argue for a taxonomy of intermediate states—dead-end, misleading, mixed, critical, and redundant-viable—as a useful lens for interpreting ToT search.

2 Related Work

Chain-of-thought and tree-of-thought reasoning. CoT prompting elicits intermediate reasoning steps and improves performance on arithmetic, symbolic, and commonsense reasoning tasks [1]. ToT reasoning extends this idea from single trajectories to explicit search over intermediate thoughts [2]. Follow-up work has explored modular controllers, memory, checkers, and graph-structured variants of thought search [3, 4]. Our work does not propose a new search controller; instead, it asks how to analyze the states produced by such controllers.

Efficiency-oriented ToT methods. Because branching search can be expensive, much ToT work focuses on pruning. Semantic similarity-based dynamic pruning, for example, removes semantically similar branches under the assumption that they are redundant [5]. Such approaches are valuable, but they typically use local or surface-level proxies. We instead measure downstream utility directly: if a branch is reached, how often does it still lead to success?

Reasoning interpretability and resampling. Recent work has argued that single reasoning traces are insufficient for understanding model reasoning. Resampling from partial traces offers a way to study distributions over possible continuations rather than a single observed path. Our work applies this idea to explicit search trees, where the controller already exposes intermediate nodes, parent-child links, and branch ranks.

3 Method

3.1 Controlled ToT-Style Search

Both tasks use a controlled ToT-style design. Candidate thoughts are generated by deterministic or dataset-derived mechanisms, and the LLM is used primarily as a *ranker* over legal candidates. This is an intentional methodological choice. Fully generative ToT conflates several sources of error: invalid thought generation, parsing failures, ranking mistakes, and true state utility. By controlling candidate generation, we isolate the question we care about: among valid candidate states, which preserve downstream success?

In Game24, all legal arithmetic operations over the remaining numbers are generated symbolically. In crosswords, candidate fills come from a dataset-derived candidate bank and are filtered by crossing-letter compatibility. In both cases, a locally served LLaMA-family model through Ollama ranks candidate continuations, and the search expands the selected top branches.

Figure 1 summarizes the experimental pipeline. The core methodological choice is that the LLM ranks candidate thoughts, while candidate generation and validity checking are controlled outside the model. This makes the experiment a diagnostic of search-state utility rather than a test of whether the model can freely generate valid actions.

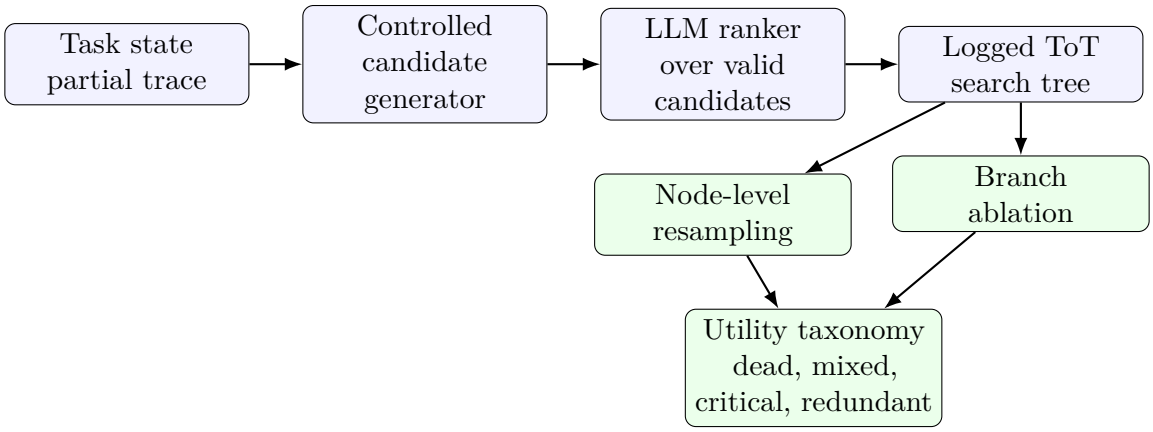

Figure 1: Experimental pipeline. We use controlled candidate generation to ensure valid actions, LLM ranking to guide search, full tree logging to expose intermediate states, and resampling/ablation to estimate utility and causal role.

3.2 Search Tree Formalization

We represent a ToT search as a rooted directed tree

$$T = (V, E),$$

where each node $v \in V$ corresponds to an intermediate reasoning state and each edge $(u, v) \in E$ corresponds to one selected continuation from parent u to child v.

Each node has a state representation

$$x_v = (\tau_v, r_v),$$

where τ_v is the partial reasoning trace and r_v is the task state. In Game24, r_v is the multiset of remaining numbers. In crossword search, r_v is the partial grid and slot assignments.

Given a node v, define its downstream utility as

$$p(v) = P(\text{success} \mid x_v),$$

where success is defined by the task: reaching 24 for Game24 or reconstructing the target crossword grid.

3.3 Node-Level Resampling

Since $p(v)$ is not directly observable, we estimate it through resampling. We restart the search from x_v a total of M times. Let $Y_i(v) = 1$ if resampling trial i from node v reaches a correct terminal state, and 0 otherwise. The empirical node utility is

$$\hat{p}(v) = \frac{1}{M} \sum_{i=1}^{M} Y_i(v).$$

We also record the mean number of successful terminal leaves found during resampling:

$$R(v) = \frac{1}{M} \sum_{i=1}^{M} L_i(v),$$

where $L_i(v)$ is the number of successful leaves in resampling trial i. This matters because binary success and solution mass are different. A node may not be uniquely necessary for a solution, but it may preserve many successful continuations.

3.4 Operational Node Categories

We assign node categories using the same operational thresholds across tasks:

$$\begin{aligned}
\texttt{dead_end} &\iff \hat{p}(v) \leq 0.05, \\
\texttt{misleading} &\iff b(v) \leq 2 \text{ and } \hat{p}(v) \leq 0.25, \\
\texttt{redundant_viable} &\iff \hat{p}(v) \geq 0.75 \text{ and } R(v) > 1.5, \\
\texttt{critical} &\iff \hat{p}(v) \geq 0.75 \text{ and not redundant-viable}, \\
\texttt{mixed} &\iff \text{otherwise.}
\end{aligned}$$

Here $b(v)$ denotes the local branch rank assigned by the LLM, with rank 1 being the most preferred candidate. In the implementation, `dead_end` is checked before `misleading`; therefore, a rank-1 node with exactly zero success is counted as a dead-end rather than a misleading node. This ordering is important when interpreting small misleading counts.

These labels are policy-relative. A node is not universally dead or critical; it is dead or critical under a given model, candidate generator, branch factor, temperature, and resampling budget.

Because all final experiments use $M = 10$ resampling trials per node, empirical success rates are coarse Monte Carlo estimates with increments of 0.1. Under this budget, the dead-end threshold corresponds to 0/10 observed successful continuations, while the critical threshold corresponds to at least 8/10 successful continuations. We therefore interpret categories as operational labels under a fixed search policy, not as intrinsic mathematical properties of the state.

Temperature also changes the meaning of repeated trials. At $T = 0.0$, resampling primarily tests reachability under a nearly deterministic ranking policy; repeated trials are expected to be stable unless downstream implementation details introduce variation. At $T = 0.2$, resampling estimates success under a mildly stochastic continuation policy. This is why temperature is useful as an instability probe: states that are stable under deterministic search but unstable under mild stochasticity appear as mixed or misleading under the stochastic condition.

3.5 Branch Ablation

Node-level resampling estimates utility conditional on reaching a node. Branch ablation asks whether that branch matters to the original tree. For a selected branch e, we compare normal search against an ablated search in which e is blocked.

Let Z_i indicate whether baseline trial i solves the puzzle, and $Z_i^{(-e)}$ whether trial i solves the puzzle

after branch e is blocked. We define

$$\hat{s}_{\text{base}}(e) = \frac{1}{M}\sum_{i=1}^{M} Z_i, \quad \hat{s}_{\text{abl}}(e) = \frac{1}{M}\sum_{i=1}^{M} Z_i^{(-e)},$$

and the solve-rate effect

$$\Delta_{\text{solve}}(e) = \hat{s}_{\text{base}}(e) - \hat{s}_{\text{abl}}(e).$$

Positive Δ_{solve} means removing the branch hurts binary solve rate. Negative values mean removing the branch helps.

We also measure successful-leaf effect:

$$\Delta_{\text{leaf}}(e) = \hat{\ell}_{\text{base}}(e) - \hat{\ell}_{\text{abl}}(e),$$

where $\hat{\ell}$ is the average number of successful leaves. This is a proxy for solution mass and redundancy. A branch can have $\Delta_{\text{solve}} \approx 0$ but $\Delta_{\text{leaf}} > 0$, meaning that it does not determine whether the puzzle is solved at least once but does preserve many successful futures.

4 Tasks and Experimental Setup

4.1 Game24

Game24 asks the solver to combine four numbers using $+, -, \times, \div$ to obtain 24. Every complete solution uses exactly three arithmetic operations, so the maximum internal depth is shallow and interpretable. We use deterministic arithmetic child generation and LLM ranking. The 100-puzzle node-resampling experiment uses:

- 100 Game24 puzzles,
- two temperatures, $T = 0.0$ and $T = 0.2$,
- maximum search depth 3,
- branch factor 4,
- selected nodes from depths 1 and 2,
- up to 8 selected nodes per puzzle,
- 10 resampling trials per node.

The resulting study contains 1,600 node summaries and 16,000 resampling trials. The Game24 node-resampling run used the local model recorded in the metadata (`llama3:latest`); later ablation and crossword runs used the locally configured LLaMA-family model in the corresponding scripts.

4.2 Game24 Branch Ablation

The comprehensive Game24 branch ablation focuses on depth-1 candidate branches. It covers 100 puzzles under two temperatures and contains 792 ablation rows. The count is slightly below $100 \times 2 \times 4 = 800$ because a few puzzles produce fewer than four valid depth-1 candidates.

For each candidate branch, the ablation script:

1. estimates node utility through resampling,
2. runs baseline full-tree search trials,

Experiment	Node summaries	Trials	Tree logs	Avg. success leaves
Game24 resampling	1,600	16,000	200	2.935
Game24 ablation	792	–	–	–
Crossword branch-diverse	387	3,870	40	0.575
Crossword high-score-path	248	2,480	40	0.525

Table 1: Final experimental scale. Game24 provides broad arithmetic coverage; crosswords provide deeper, lower-yield constraint search.

Experiment	Model	Temps.	Node selection	Search limits	Trials/node
Game24 resampling	llama3:latest	0.0, 0.2	depths 1–2, up to 8 nodes/puzzle	branch 4, max depth 3	10
Game24 ablation	llama3.2	0.0, 0.2	depth-1 candidate branches	branch 4, max depth 3	10
Crossword branch-diverse	llama3.2	0.0, 0.2	depth/rank diverse, terminal dead included	branch 4, frontier-limited beam search	10
Crossword high-score path	llama3.2	0.0, 0.2	high path-score internal states	branch 4, frontier-limited beam search	10

Table 2: Experimental configuration. Game24 and crossword experiments share the same resampling logic but differ in candidate generation, tree depth, and node-selection policy.

3. reruns full-tree search while blocking that branch,
4. records solve-rate drop and successful-leaf drop.

4.3 Crossword-Style Fill Search

The crossword experiments provide a deeper, constraint-propagating contrast to Game24. We construct an expanded crossword-style benchmark from Tree-of-Thought mini-crossword clue/answer material. The final crossword experiments use 20 medium 11x11 puzzles with up to 40 fill slots per puzzle. Candidate fills are drawn from a candidate bank and filtered by crossing-letter compatibility. The LLM ranks compatible candidate fills.

The benchmark is controlled by design. Each puzzle instance consists of a target grid, a set of across/down slots derived from the grid, clue text for each slot, and a candidate bank that contains the target answer plus distractor candidates of the same length. During search, the next slot is chosen using constraint information, candidate words are filtered against already-filled crossing letters, and the LLM ranks only the remaining compatible candidates. Therefore, the crossword experiment should be interpreted as controlled crossword-style fill search, not as fully open-ended crossword solving from raw clues alone.

We run two crossword selection regimes:

- **High-score-path**: selects high-path-score internal states, mostly from preferred rank-1 trajectories.
- **Branch-diverse**: selects across depth and branch rank, including terminal dead nodes, to expose useless and misleading thoughts.

The branch-diverse run contains 387 node summaries and 3,870 resampling trials. The high-score-path support run contains 248 node summaries and 2,480 resampling trials.

The main text reports means and counts for readability. The final analysis artifacts also include standard errors and confidence intervals for the key grouped summaries. Because ablation rows are

Temp.	Dead-end	Critical	Redundant viable	Mixed	Misleading
0.0	598	101	101	0	0
0.2	564	96	84	47	9
Total	1,162	197	185	47	9

Table 3: Game24 node categories by temperature. Mild stochasticity exposes mixed and misleading nodes that deterministic search hides.

Temp.	Depth	Nodes	Avg. success	Avg. successful leaves
0.0	1	396	0.3889	0.7854
0.0	2	404	0.1188	0.1411
0.2	1	397	0.3861	0.7327
0.2	2	403	0.1052	0.1223

Table 4: Game24 utility by depth. In shallow arithmetic search, early states often determine whether successful futures remain accessible.

branch-level observations clustered by puzzle and temperature, we interpret category-level ablation averages descriptively rather than as independent-sample causal estimates.

5 Game24 Results

5.1 Tree-Level Behavior

Across 200 Game24 trees (100 puzzles at two temperatures), 159 contain at least one successful terminal leaf. The average tree has 82.365 nodes, maximum depth 3, 61.685 terminal nodes, and 2.935 successful leaves. This means the setting is neither saturated nor impossible: most trees solve at least once, but there is enough failure structure to analyze which nodes matter.

5.2 Node Categories and Temperature

Table 3 shows that Game24 node utility is strongly polarized. At $T = 0.0$, all nodes fall into three categories: dead-end, critical, or redundant-viable. At $T = 0.2$, mixed and misleading nodes appear.

This makes temperature a useful instability probe. Deterministic search produces sharply separated states; mild stochasticity reveals nodes whose utility is unstable or brittle under small policy perturbations.

5.3 Depth Is Strong, but Task-Specific

Depth is the strongest aggregate predictor in Game24 (Table 4). Depth-1 states have average success around 0.39, while depth-2 states have average success around 0.11–0.12.

This does not mean "early nodes are always better." It means that in a three-operation arithmetic task, the first operation often opens or closes large regions of the remaining search space. Depth is therefore a task-geometry signal, not a universal law.

Temp.	Rank	Nodes	Avg. success	Avg. successful leaves
0.0	1	496	0.1613	0.2440
0.0	2	108	0.4074	0.8333
0.0	3	100	0.3900	0.8700
0.0	4	96	0.4062	0.7292
0.2	1	497	0.1646	0.2521
0.2	2	106	0.3802	0.7538
0.2	3	100	0.3790	0.7210
0.2	4	97	0.3680	0.6485

Table 5: Game24 utility by local branch rank. The raw rank effect is striking, but partially confounded by depth; the safest conclusion is that local rank alone is not a reliable global utility proxy.

5.4 Local Rank Is Not a Sufficient Utility Proxy

The raw Game24 rank table (Table 5) shows that rank-1 nodes have much lower average success than ranks 2–4. However, this must be interpreted carefully: the rank-1 bucket contains many depth-2 nodes, and depth-2 nodes are generally low utility. Within depth 1 alone, branch ranks are much closer. At $T = 0.0$, depth-1 rank-1 nodes average 0.370 success, while non-rank-1 depth-1 nodes average about 0.395. At $T = 0.2$, depth-1 rank-1 nodes average 0.413, while non-rank-1 depth-1 nodes average about 0.377.

The defensible claim is not "rank 1 is always bad." The defensible claim is that local rank alone is insufficient: state position, task structure, and remaining search all matter.

5.5 Branch Ablation Validates the Taxonomy

The ablation study asks whether node-utility categories matter to the whole tree. The answer is yes, but the effect depends on the metric.

Overall, binary solve rate barely changes:

$$\text{baseline solve rate} = 0.6806, \quad \text{ablated solve rate} = 0.6822, \quad \Delta_{\text{solve}} = -0.0016.$$

If we only measured whether at least one solution remains, ablation would appear neutral. But successful solution mass decreases:

$$\text{baseline successful leaves} = 2.4283, \quad \text{ablated successful leaves} = 2.2655, \quad \Delta_{\text{leaf}} = 0.1628.$$

Table 6 shows that the category-level effects are much richer.

Dead-end branches often consume branch budget: removing them improves solve rate and successful leaves on average. Critical branches have positive solve-rate drop, meaning they matter for binary success. Redundant-viable branches have the largest successful-leaf drop, meaning they preserve solution mass even when another solution remains. This validates the taxonomy causally within the controlled search policy.

Category	n	Node succ.	Base solve	Abl. solve	Solve drop	Leaf drop
dead_end	499	0.0000	0.5244	0.5930	-0.0685	-0.3128
critical	113	0.9735	0.9779	0.8646	0.1133	0.3938
redundant_viable	139	0.9986	0.9835	0.8475	0.1360	1.7791
mixed	31	0.3710	0.7903	0.7677	0.0226	-0.2032
misleading	10	0.1100	0.5600	0.5100	0.0500	-0.0500

Table 6: Game24 branch ablation by node category. Dead-end branches are neutral or beneficial to remove on average; critical and redundant-viable branches hurt when removed, especially in successful-leaf mass.

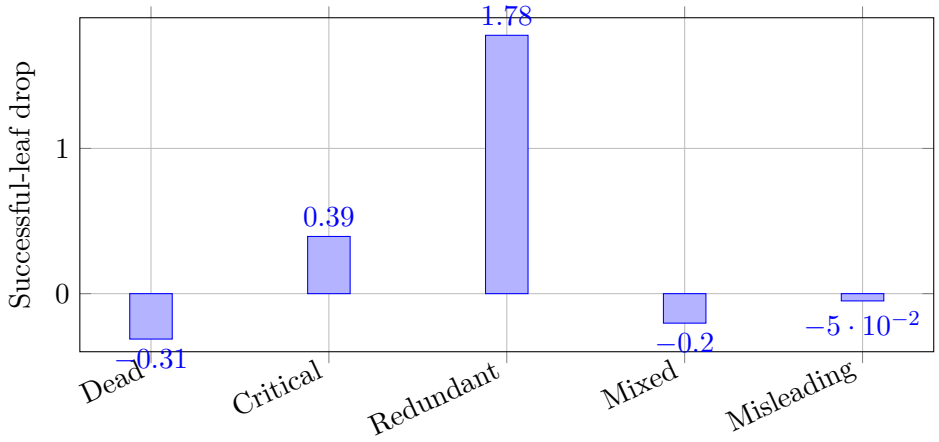

Figure 2: Successful-leaf drop under Game24 branch ablation. Redundant-viable branches preserve many successful futures even when binary solve rate is not destroyed.

5.6 Qualitative Game24 Examples

Table 7 gives representative examples. The first two show that local rank can mislead: in `1 1 4 6`, the rank-1 move `1+1=2` is a dead-end under resampling, while in `1 1 11 11`, the rank-4 move `11+11=22` is critical. The next two show why ablation matters: removing a dead-end branch can improve search, while removing a redundant-viable branch can preserve binary solve rate but reduce solution mass.

6 Crossword Results

6.1 Tree-Level Behavior

Crossword search is a harder, deeper, lower-yield setting. The branch-diverse crossword run has 40 tree logs, average max depth 28.975, and only 23 of 40 trees contain at least one success. The average number of successful leaves is 0.575. The high-score-path run has 40 trees, average max depth 27.8, and 21 successful trees with average 0.525 successful leaves.

This confirms that crosswords are not merely "more Game24." They are a different search geometry.

Puzzle	Thought	Category	Utility	Interpretation
1 1 4 6	$1 + 1 = 2$	dead_end	0.0	Locally top-ranked but no successful futures.
1 1 11 11	$11 + 11 = 22$	critical	1.0	Lower-ranked but reliably preserves success.
1 2 3 4	$1 + 3 = 4$	dead_end	0.0	Ablation improves solve rate from 0.0 to 1.0.
2 2 4 6	$4 \times 6 = 24$	redundant_viable	1.0	Ablation leaves solve rate at 1.0 but reduces successful leaves from 7.0 to 1.0.

Table 7: Representative Game24 examples illustrating dead-end, critical, and redundant-viable behavior.

Setting	Trees	Success trees	Avg. nodes	Avg. max depth	Avg. success leaves
Game24	200	159	82.365	3.000	2.935
Crossword branch-diverse	40	23	37.525	28.975	0.575
Crossword high-score-path	40	21	36.000	27.800	0.525

Table 8: Tree-level difficulty by task. Crossword search is deeper and lower-yield than Game24.

6.2 Category Distribution and Temperature

Table 9 shows the crossword category distribution. The branch-diverse run exposes many more dead-end alternatives than the high-score-path run, because it intentionally samples across branch rank and includes terminal dead nodes.

As in Game24, stochasticity acts as an instability probe. At $T = 0.0$, the crossword categories are polarized into critical and dead-end nodes. At $T = 0.2$, mixed and misleading states appear.

6.3 Depth-Utility Curve

The crossword depth curve differs sharply from Game24. In the branch-diverse run, depth-1 nodes have low success: 0.1806 at $T = 0.0$ and 0.1458 at $T = 0.2$. But selected deeper surviving states become much more reliable. At depths 24–28, success is typically around 0.83–0.93, and among selected depth-32 and depth-36 nodes it reaches 1.0. The deepest bins have small sample sizes, so this should be interpreted cautiously.

This is one of the most important cross-task findings. In Game24, early arithmetic states dominate. In crossword search, wrong early fills often die immediately, while deeper surviving states are filtered by constraints and closer to completion.

6.4 Branch Rank and Constraint Collapse

Crossword rank behaves differently from Game24. In the branch-diverse run, rank-1 nodes are often useful, while lower-ranked candidates are overwhelmingly dead. But rank 1 is not perfect: it still contains 46 dead-end, 17 mixed, and 13 misleading nodes.

Run	Temp.	Critical	Dead-end	Mixed	Misleading
Branch-diverse	0.0	120	84	0	0
Branch-diverse	0.2	89	64	17	13
High-score path	0.0	117	24	0	0
High-score path	0.2	78	6	10	13

Table 9: Crossword node categories by run and temperature. Branch-diverse selection reveals more dead alternatives; stochasticity introduces mixed and misleading states.

Temp.	Depth	n	Avg. success	Avg. leaves	Mean resampled nodes
0.0	1	72	0.1806	0.1806	10.8194
0.0	4	18	0.7222	0.7222	37.2222
0.0	16	17	0.7647	0.7647	25.1765
0.0	24	15	0.8667	0.8667	17.6667
0.0	32	13	1.0000	1.0000	9.0000
0.0	36	3	1.0000	1.0000	5.0000
0.2	1	72	0.1458	0.1458	9.6375
0.2	4	17	0.6294	0.6294	33.8529
0.2	16	14	0.7500	0.7500	24.7500
0.2	24	12	0.9333	0.9333	18.1833
0.2	32	10	1.0000	1.0000	9.0000
0.2	36	2	1.0000	1.0000	5.0000

Table 10: Selected branch-diverse crossword depth results. Deeper surviving states become highly reliable, but this reflects both constraint survival and less remaining search.

This demonstrates that local rank is task-dependent. In crossword fill search, crossing constraints make many lower-ranked alternatives collapse immediately. In Game24, arithmetic alternatives can be less locally separable, and rank is more confounded by depth.

6.5 Qualitative Crossword Examples

The crossword examples illustrate the constraint geometry. In `tot_medium11x11_000`, the lower-ranked early fill `1A PAREL` immediately becomes a terminal dead node with zero success. In the same puzzle, a deep state after `24D NUMPS` at depth 32 has success rate 1.0. The grid has survived many crossing constraints, and only a small amount of search remains.

A separate example from `tot_medium11x11_007` shows a top-ranked but misleading node: `1A COULD` has resampled success 0.2. This illustrates that even when rank is more aligned in crosswords than in Game24, local rank still fails often enough to justify resampling.

7 Cross-Task Analysis

The Game24 and crossword results should not be collapsed into a single depth story. They reveal different search geometries.

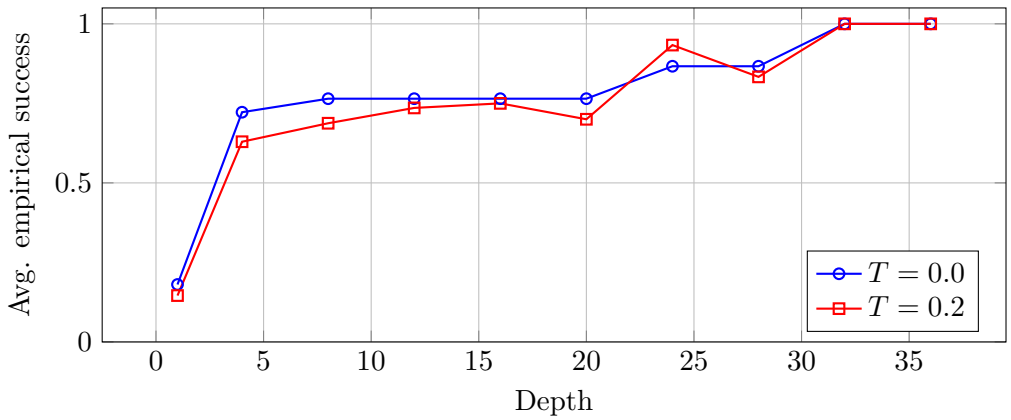

Figure 3: Crossword branch-diverse depth-utility curve. Unlike Game24, selected deeper surviving states become more reliable as constraints accumulate and remaining search shrinks.

Branch rank	Critical	Dead-end	Mixed	Misleading
1	207	46	17	13
2	0	40	0	0
3	2	30	0	0
4	0	32	0	0

Table 11: Crossword branch-diverse categories by branch rank. Rank aligns better with utility than in Game24, but top-ranked nodes still include failures.

Game24: shallow arithmetic geometry. Game24 solutions have exactly three operations. The first operation can quickly create or destroy a useful intermediate quantity. This explains why depth-1 states are much more useful than depth-2 states on average. Game24 also has substantial redundancy: once a useful intermediate number is created, multiple arithmetic continuations may still reach 24. This is why `redundant_viable` nodes are common and why successful-leaf mass is such an important metric.

Crosswords: constraint-propagating geometry. Crossword search is deeper and lower-yield. Wrong early fills often terminate immediately because they make future crossing slots impossible. Surviving deeper states become more reliable because they have passed many compatibility checks and have fewer remaining decisions. This explains why branch-diverse crosswords have many early dead-end nodes and highly reliable selected deep nodes. Unlike Game24, the crossword runs contain no redundant-viable nodes under our thresholds, suggesting that successful crossword completions are narrower and more constraint-tight.

Unified interpretation. The same method reveals different task geometries. Node utility is not a universal function of depth or rank. It is shaped by task structure, candidate generation, branch factor, model ranking, constraints, and remaining search. This is the strongest reason to study intermediate-state utility directly rather than inferring it from local scores or final accuracy.

Puzzle	Thought	Category	Utility	Interpretation
tot_medium11x11_000	1A PAREL	dead_end	0.0	Wrong early fill dies immediately.
tot_medium11x11_000	24D NUMPS	critical	1.0	Deep surviving state is reliable.
tot_medium11x11_007	1A COULD	misleading	0.2	Top-ranked fill has low downstream success.

Table 12: Representative crossword examples. The task exhibits immediate wrong-branch collapse and reliable deep surviving states.

8 Discussion

8.1 Why This Is Not Merely Running ToT

The project's contribution is not that it runs ToT on two tasks. The contribution is a shift in unit of analysis. Instead of measuring only final accuracy, we measure the conditional utility of intermediate search states. This lets us distinguish:

- states that look plausible but have no future,
- states that reliably preserve success,
- states that preserve multiple successful futures,
- states that are unstable under stochasticity,
- and states whose importance appears only under ablation.

This is especially important because final-answer accuracy hides search-structure loss. A puzzle may remain solvable after ablating a branch, but the number of successful leaves can collapse. That is a real loss of robustness or solution diversity.

8.2 Temperature as an Instability Probe

Temperature reveals brittle states. In both Game24 and crosswords, deterministic search produces polarized categories. Mild stochasticity introduces mixed and misleading states. This suggests that resampling under multiple search policies can distinguish stable utility from policy-sensitive utility.

8.3 Successful-Leaf Mass as a Search Metric

Binary solve rate is coarse. It treats a tree with one successful leaf and a tree with seven successful leaves as equivalent. Our ablation study shows that this misses important structure. Redundant-viable Game24 branches often preserve successful-leaf mass even when binary solve rate remains unchanged. Therefore, successful-leaf count acts as a proxy for solution mass, redundancy, and robustness.

8.4 Controlled Candidate Generation as Strength and Limitation

Our system does not use fully open-ended LLM thought generation. This is a limitation for generality, but also a strength for interpretation. Controlled candidate generation ensures that candidate moves or fills are legal, so measured failures are not simply invalid syntax or malformed outputs. The study isolates a cleaner question: whether valid intermediate states preserve downstream success under an LLM-ranked search policy.

9 Limitations

The limitations below are also part of the experimental design. We deliberately used controlled candidate generation to isolate intermediate-state utility from invalid action generation. This gives cleaner causal and diagnostic conclusions, but it means the results should be read as evidence about controlled ToT-style search with LLM ranking, not as evidence about arbitrary free-form thought generation.

First, the LLM is primarily a ranker/evaluator, not a fully open-ended thought generator. Game24 candidates are deterministic arithmetic operations, and crossword candidates come from a candidate bank with crossing constraints. Future work should test fully generative ToT where the model proposes candidates from scratch.

Second, category labels are policy-relative. A node categorized as dead-end under one model, branch factor, temperature, or candidate set may not be dead under another. This is scientifically important: the method estimates utility under a specified search policy.

Third, Game24 rank analysis is depth-confounded. The raw branch-rank table should not be interpreted as proving that rank 1 is intrinsically worse. The safer conclusion is that local rank alone is not enough, and depth/state position must be considered.

Fourth, the crossword branch-diverse selection intentionally oversamples across branch rank and includes terminal dead nodes. This is useful for studying the taxonomy of useful and useless thoughts, but its category distribution should not be interpreted as the natural distribution of all nodes in an unconstrained search.

Fifth, deepest crossword bins have small sample sizes. Their high success rates should be interpreted as evidence about selected surviving states, not as a universal claim that all deep crossword states are reliable.

Finally, the Game24 ablation evidence is strongest for depth-1 branches. Extending ablation to deeper subtrees and to crosswords is an important future step.

Ablation rows are branch-level observations and are clustered by puzzle and temperature. We therefore use the ablation results as descriptive causal stress tests within the controlled search policy, rather than as independent-sample statistical estimates over arbitrary puzzles.

10 Future Work

Fully generative ToT. The most direct extension is to allow the LLM to propose candidate thoughts rather than only ranking controlled candidates. Resampling would then measure both

generation quality and downstream state utility.

Learning pruning policies from utility labels. Node resampling produces labels that could train a pruning model. Instead of pruning by local rank or semantic similarity, a future controller could learn to predict dead, critical, mixed, and redundant-viable states from local features.

Comparison with semantic similarity pruning. A key future question is whether semantic similarity aligns with downstream utility. Some semantically similar branches may have different causal roles, while semantically different branches may be functionally redundant.

Larger crossword corpora. The crossword benchmark should be extended to historical or XD-style crossword corpora. This would test whether the constraint-propagation geometry observed here generalizes to less controlled crossword data.

Crossword ablation. The Game24 ablation validates the utility taxonomy causally. A comparable crossword ablation would test whether wrong early fills merely die locally or also affect global search capacity.

11 Conclusion

We introduced Resampled Thought Trees, a framework for measuring the downstream utility of intermediate states in Tree-of-Thought search. Rather than evaluating only completed trajectories or local branch scores, we treat intermediate thoughts as search states with measurable conditional futures.

On Game24, resampling reveals a polarized arithmetic landscape of dead-end, critical, and redundant-viable states. Ablation validates these categories: dead-end branches can waste branch budget, critical branches affect binary success, and redundant-viable branches preserve solution mass. On crossword-style fill search, the same framework reveals a different geometry: wrong early fills often collapse immediately, while deeper surviving states become reliable because crossing constraints have pruned invalid alternatives and fewer decisions remain.

The overall conclusion is that intermediate thoughts are not just earlier or later, and not just high- or low-ranked. They have task-dependent utility. Understanding ToT search requires measuring those utilities directly.

A Appendix: Operational Category Definitions

The node categories are assigned in order:

1. `dead_end`: $\hat{p}(v) \leq 0.05$.
2. `misleading`: branch rank ≤ 2 and $\hat{p}(v) \leq 0.25$, excluding dead-end nodes.
3. `redundant_viable`: $\hat{p}(v) \geq 0.75$ and $R(v) > 1.5$.
4. `critical`: $\hat{p}(v) \geq 0.75$, excluding redundant-viable nodes.
5. `mixed`: all remaining nodes.

This ordering explains why some high-ranked zero-success nodes are classified as dead-end rather than misleading.

B Appendix: Additional Figures

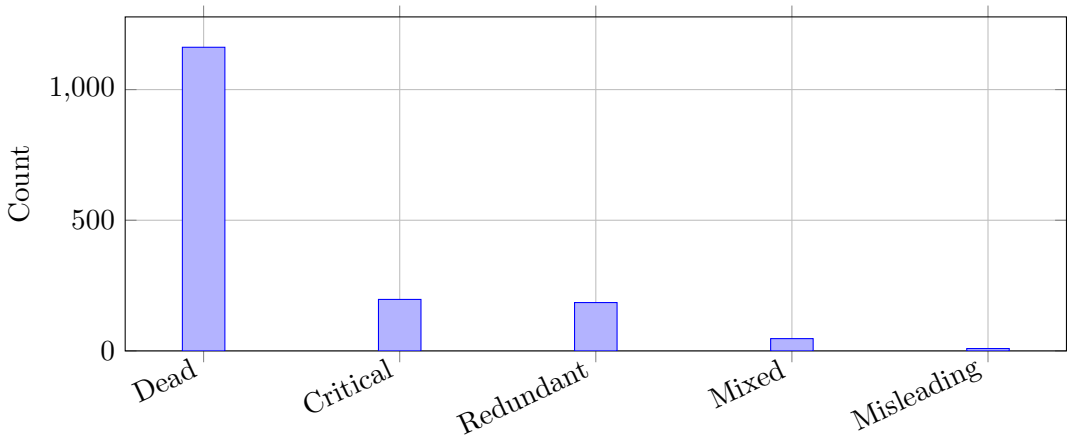

Figure 4: Overall Game24 node-category distribution across both temperatures.

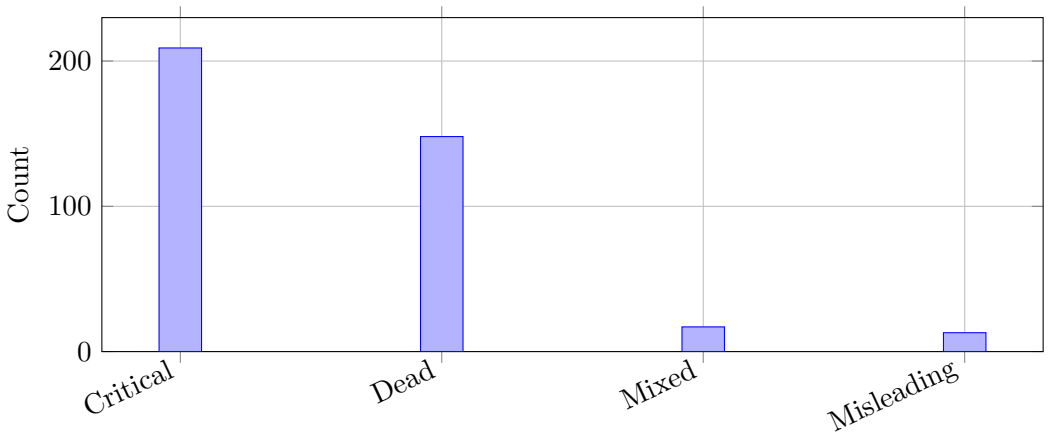

Figure 5: Crossword branch-diverse node-category distribution. Unlike Game24, no redundant-viable nodes appear under our thresholds.

References

[1] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

[2] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 2023.

[3] Jinpeng Long. Large language model guided tree-of-thought. *arXiv preprint arXiv:2305.08291*, 2023.

[4] Maciej Besta, Nils Blach, Ales Kubicek, et al. Graph of thoughts: Solving elaborate problems with large language models. *AAAI*, 2024.

[5] Jinsol Kim, Xuan Huang, Zubair Reza, Greg Grand, Kai Zhu, and Ryan Lagasse. Chopping trees: Semantic similarity-based dynamic pruning for tree-of-thought reasoning. *arXiv preprint arXiv:2511.08595*, 2025.