of BoT will produce the most representative reasoning chain in the current iteration, thus leading to more meaningful *experience* to enhance the prompt. We verify this in the ablation study section.

To better present how BoT learns from errors and previous advice, we show in Table 3 that GPT-4 is able to avoid previous errors and produce more specific advice with the increase of iteration and eventually obtain the correct solution. In the first iteration, with the simple prompt, LLMs even make a mistake in following the task rules as the new set is wrong in step 3. After analyzing, it presents correct advice on this mistake. However, the analysis at the initial iteration is vague, such as "try other numbers and operations". After five iterations, BoT aggregates multiple such analyses, deriving a more potent prompt, making the LLMs select the right numbers 9 and 7. Also, the advice is more concrete and useful. The advice for this right selection is to increase the corresponding evaluation score. Through the continuous accumulation such *experiences*, BoT progressively refines the prompt, culminating in the direct generation of a correct solution in the 8-th iteration.

## 4.3 Ablation Study

**Table 4:** Comparison of BoT variations applied to GPT-4 on the `Game of 24` and `AQuA` Datasets.

| Experience $\mathbf{F}^{1\cdots t}$ | | Accumulation Type | | `Game of 24` | | | `AQuA` | | |
|---|---|---|---|---|---|---|---|---|---|
| Issues | Advice | Replace | Add | BoT (Best first) | BoT (Greedy) | BoT (No) | BoT (Best first) | BoT (Greedy) | BoT (No) |
| ✓ | ✓ | | ✓ | 81.2 | 83.7 | 67.1 | 78 | 81.4 | 56.2 |
| | | | | 74.7 | 78.2 | 70 | 47.3 | 56.8 | 44.9 |
| ✓ | | ✓ | | 72.8 | 74.1 | 70.2 | 52.4 | 62.7 | 46.3 |
| ✓ | | | ✓ | 69.2 | 70.7 | 67.6 | 54.1 | 60 | 40.3 |
| | ✓ | ✓ | | 74.9 | 76.9 | 72.7 | 68.3 | 74.2 | 71.9 |
| | ✓ | | ✓ | 77.9 | 80 | 72.4 | 73.6 | 77 | 64.1 |

***Experience* consistently leads to thought revision, but too much can have the opposite effect.** When the prompt accumulates issues and advice by the "adding" type, both aggregation strategies can lead to high solve rates. Maintaining a complete experience is important for revising thoughts, especially for the `AQuA` dataset, which includes wider mathematical reasoning problems. However, BoT (No), which does not perform aggregation but directly uses all reasoning chains from generated trees, suffers the worst performance in all cases, especially when the experience accumulation type is "adding". As BoT builds 15 trees each iteration, putting them all together into a prompt may cover core information, not to mention that most such experiences are invalid or harmful.

**Advice is more important to generate thoughts than others.** In all cases of Table 4, BoT variations that embrace advice as experience achieve the top solve rate. For example, with the same "adding" type, when the experience does not contain advice, the performance drops by more than $10\%$ and $20\%$ in `Game of 24` and `AQuA`, respectively. On the contrary, including issues in the *experience* serves as an auxiliary mechanism for performance improvement. Only by cooperating issues can the BoT with advice gain the best solve rate; for example, the number grows by $4.4\%$ for BoT (Greedy) in `AQuA`.

**Greedy aggregation can be the only required choice for performance purposes.** As compared to the Best-first that selects one from existing thought chains and no aggregation that maintains all thought chains, greedy aggregation adaptively merges tree structures into one better thought chain that may not exist in the current iteration. By doing so, LLM is able to perform a more meaningful analysis on a stronger thought chain, thus producing important experiences to enhance the prompt. As shown in Table 4, once the Greedy aggregation is used, BoT improves by more than $2\%$ in all cases. In `AQuA`, containing more math problems, this number is even $10\%$. Besides, as our discussion in Fig. 4, ToT with a similar experience-driven boosting mechanism reaches $80\%$ but still lags behind the BoT. This may be attributed to the inability to execute the greedy aggregation within its singular tree structure.

## 5 Conclusion

This paper verified that a simple prompt can be enhanced by gradually accumulating error analysis on its generated thoughts to address complex tasks. We have proposed a novel framework, the Boosting of Thoughts (BoT), to implement such progressive prompt enhancement for effective thought generation