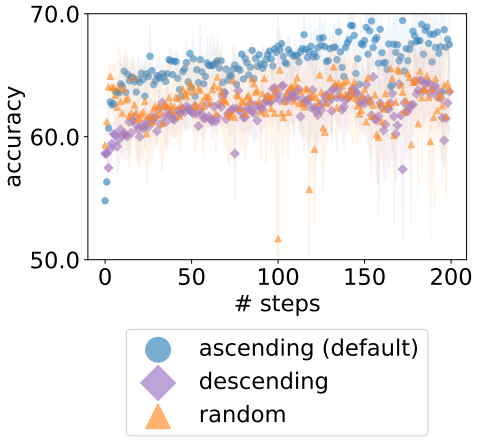
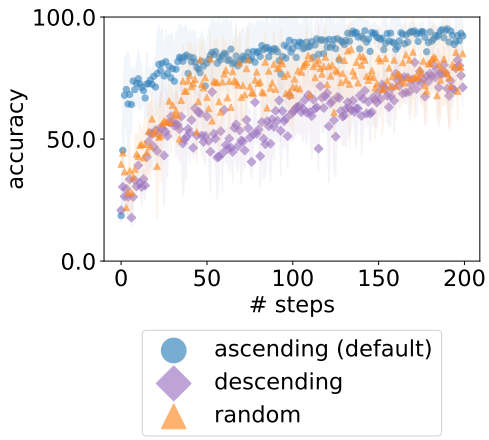
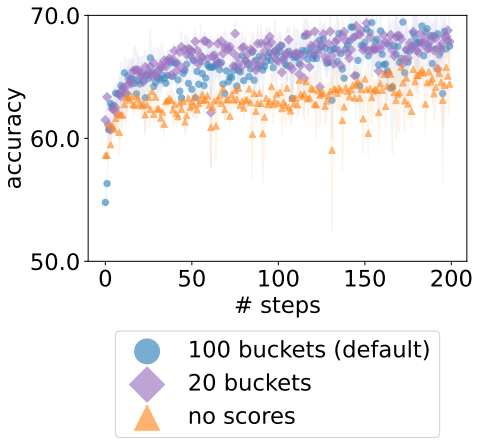


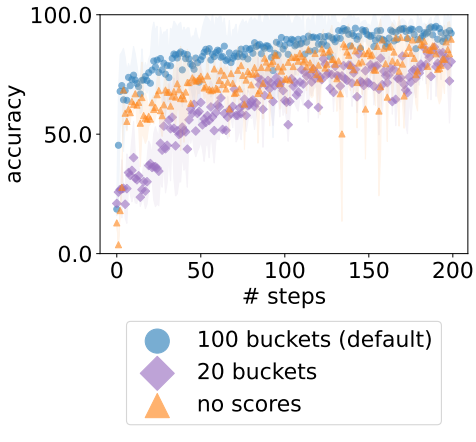
(a) instruction ordering (GSM8K)



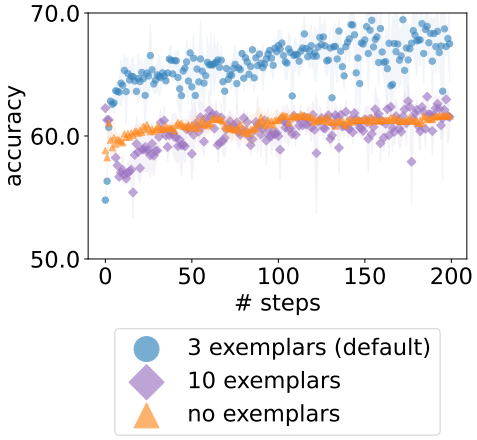
(b) instruction ordering (BBH sports_understanding)



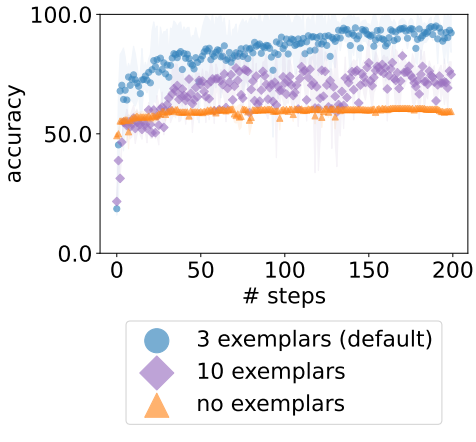
(c) instruction scores (GSM8K)



(d) instruction scores (BBH sports_understanding)

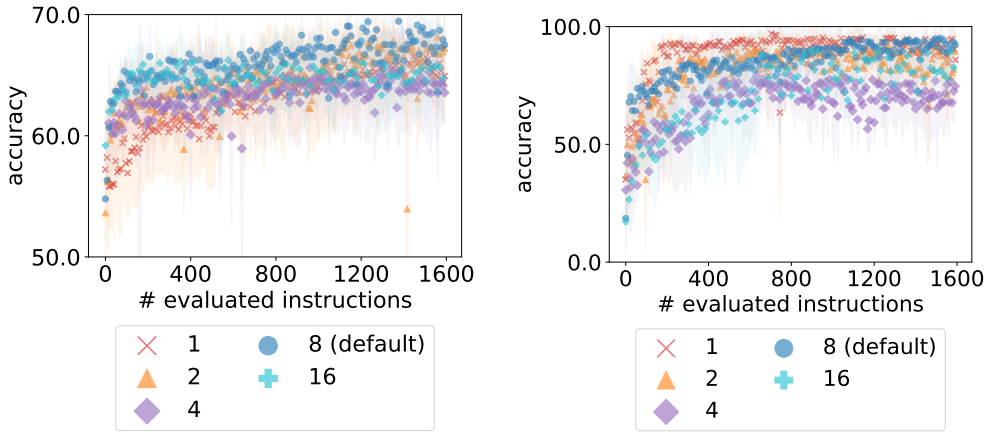


(e) # exemplars (GSM8K)



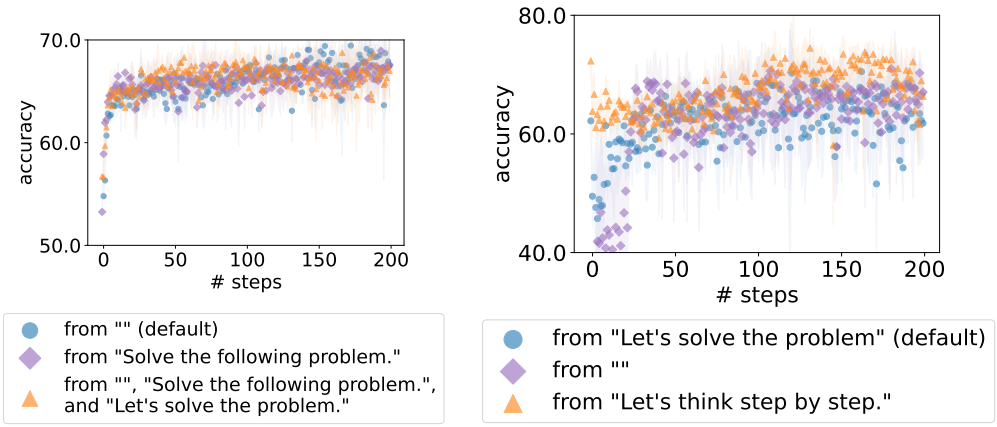
(f) # exemplars (BBH sports_understanding)

Figure 7: **Ablation studies: how each part of the meta-prompt matters.** The dots are the average values across 3 optimization repetitions, and the shaded regions represent standard deviations.



(a) GSM8K (b) BBH sports_understanding

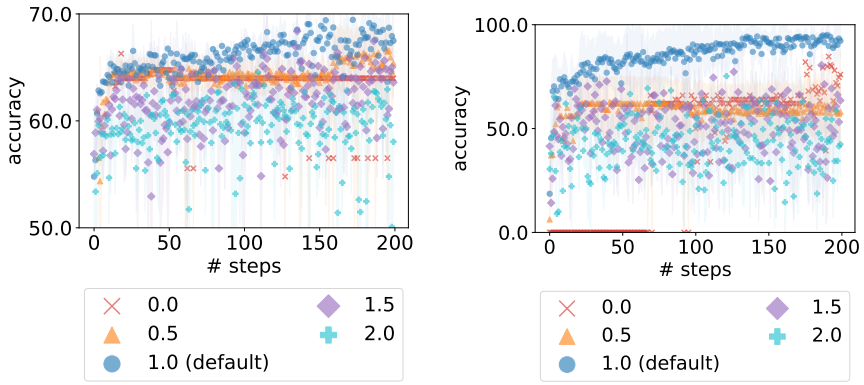
Figure 8: **Ablation studies: the number of generated instructions in each step.** The dots are the average values across 3 optimization repetitions, and the shaded regions represent standard deviations. The x-axis represents the total number of evaluated instructions through the optimization; e.g., we run 200 optimization steps when sampling 8 instructions in each step, run 400 steps when sampling 4 instructions in each step, etc.



(a) GSM8K, `text-bison` scorer, Q_begin (b) GSM8K, `PaLM 2-L` scorer, A_begin

Figure 9: **Ablation studies: the initial instructions for prompt optimization.** The dots are the average values across 3 optimization repetitions, and the shaded regions represent standard deviations.

Figure 9(b) presents the results of of `PaLM 2-L` as the scorer LLM with the following options of initial instructions: (1) "Let's solve the problem."; (2) the empty string; or (3) "Let's think step by step.". We notice that the performance differs much more with different initial instructions, especially at the beginning of the optimization. Specifically, starting from (1) leads to better generated instructions than (2) in the first 30 steps, while the instructions optimized from both (1) and (2) are worse than (3) throughout. A similar observation holds when using `PaLM 2-L` as scorer and `gpt-3.5-turbo` as optimizer for BBH tasks, by comparing the results starting from the empty string (Appendix E.2) and from "Let's solve the problem." (Appendix E.3). Taking a closer look into the optimization process of (2), we find that although both "solve the problem" and "step by step" show up in generated instructions at Step 5, it takes the optimizer LLM more steps to get rid of worse instructions presented in the meta-prompt when starting from instructions with lower accuracies. Therefore, one direction for future work is to accelerate convergence from weaker starting points.



(a) GSM8K (b) BBH sports_understanding

Figure 10: **Ablation studies: temperature of the optimizer model.** The dots are the average values across 3 optimization repetitions, and the shaded regions represent standard deviations.

Diversity per step. We evaluate the following temperatures of the optimizer LLM: {0.0, 0.5, 1.0 (default), 1.5, 2.0}. Figure 10 shows the default temperature 1.0 achieves the best performance. Specifically, optimizations with smaller temperatures (0.0 and 0.5) lack exploration and thus creativity, and the optimizer LLM often gets stuck at the same instruction for tens of steps, resulting in flat optimization curves. On the other hand, with larger temperatures (1.5 and 2.0), the optimizer LLM more often ignores the trajectory of previous instructions presented in the meta-prompt and thus lacks exploitation, therefore the optimization curve does not have a steady upward trend.

Comparison with one-step instruction generation. Our current iterative procedure runs for multiple steps and generates a new batch of solutions in each step. To validate the importance of leveraging the optimization trajectory for generating new prompts, we compare to a baseline that generates all instructions in a single step without entering into the optimization procedure. We compare these two approaches on GSM8K and BBH sports_understanding with the `PaLM 2-L-IT` optimizer. For GSM8K the scorer LLM is pre-trained `PaLM 2-L` and the initial instruction is "Let's solve the problem", and for BBH sports_understanding the scorer LLM is `text-bison` and the initial instruction is the empty string. The baseline generates 50 instructions in a single step, thus its meta-prompt only includes task exemplars, the initial instruction with its accuracy, and the same meta-instructions as our full meta-prompt for performing optimization. All the other hyperparameters remain the same.

Our results show that this one-step instruction generation performs much worse than our optimization approach. Specifically: (1) On GSM8K, the best instruction among all 50 is still "Let's solve the problem", with a 64.4 training accuracy and a 60.8 test accuracy. On the other hand, our approach (corresponding to Figure 1(a) in the main paper) found "Let's do the math!" with a 78.2 training accuracy and a 76.3 test accuracy at the 5th step by generating 8 instructions at each step. (2) Similarly, on BBH sports_understanding, the best instruction among all 50 achieved a 84.0 training accuracy and 80.0 test accuracy. This is again worse than the instruction found by our approach at Step 4, which achieved a 88.0 training accuracy and a 84.5 test accuracy.

5.4 Overfitting Analysis in Prompt Optimization

For simplicity, we do not set aside a validation set in our default setting of prompt optimization. We made this decision based on the experiments when a validation set is present.

Overfitting may result in training accuracy being much higher than the validation/test accuracy. It is difficult to avoid overfitting, but overfitting is less harmful when each candidate solution (natural language instruction in the prompt optimization context) overfits to a similar extent. In this case, a higher training accuracy solution still achieves a higher validation/test accuracy, and one can adopt solutions with the highest training accuracies as the final result. Figure 11 shows this is the case for OPRO in prompt optimization: when setting aside a validation set with the same size as the training