Your concern: Why is EoH not used for comparison for the FSSP problem?

My response:We used EoH for comparison on the FSSP problem, as shown in Table 3 and Table 9 in the appendix. We didn't use ReEvo for comparison on FSSP since ReEvo has no FSSP experiments or prompts, making it impossible to compare PoH with ReEvo on FSSP.

Your concern: In experimental settings were all algorithms able to solve all instances? If not, how was the optimality gap considered?I assume the LLM is used as the transition function. In this case, how is the validity of heuristics generated calculated? That is, to make sure the LLM output is correct code, i.e., the actions generated are valid (belong to action space?).

My response: Thank you for raising this important concern. We agree that purely greedy selection of actions with the highest immediate local rewards could limit exploration and risk missing globally optimal trajectories. However, PoH explored only 60 heuristics, while EoH generated 400 heuristics over 20 generations × 20 iterations. The better performance of the heuristic obtained by PoH demonstrates that PoH's greedy strategy can efficiently explore the heuristic space. In the future research, we will actively explore random sample strategies and other strategies as well.

Your concern:In experimental settings were all algorithms able to solve all instances? If not, how was the optimality gap considered?I assume the LLM is used as the transition function. In this case, how is the validity of heuristics generated calculated? That is, to make sure the LLM output is correct code, i.e., the actions generated are valid (belong to action space?).

My response: PoH first calls the LLM to generate (initialize) a heuristic based on the prompt. Then, we process the code generated by the LLM to obtain the generated distance matrix update function. (The LLM's output is text, and we use Python's re.findall function to find the corresponding code section, which is then converted into an imported module using the importlib.import_module function.) Subsequently, we combine the generated distance matrix update function with GLS to solve TSP instances and evaluate the performance of the generated heuristic (the reward for the current heuristic). The action generation part involves the LLM analyzing the current and past heuristic (states), including their code, descriptions, and evaluation results, to provide improvement suggestions for the current heuristic, aiding in generating a better one next time. Therefore, since the heuristic designed by PoH can solve all instances in most cases, if the generated code contains errors (code errors), the reward is 0. In the next action generation part, LLMs may notice this abnormal situation (reward is 0) and carry out self - reflection. They will review their previous outputs and utilize their strong language domain knowledge and powerful code generation ability to obtain better heuristic and guide their continuous improvement.

Your concern: 3.2 reward function is defined as 1 minus the percentage... the optimal solution for an instance how is this measured? How can we obtain this optimal solution in cases where these values are unknown for vanilla MCTS?

My response: Regarding the reward function setup, we use current baseline algorithms (e.g., Concorde or LKH for the TSP) on a pre-generated training set to get the shortest distance (optimal value). During training, we replace GLS's update distance matrix function with code generated by the LLM, getting a distance value (evaluation value). To keep the reward between 0 and 1, and ensure higher rewards are better, we apply the formula: $R = 1 - \frac{O_{st} - O_{best}}{O_{best}}$. Thus, the optimal solution is the optimal value obtained by the pre-selected baseline algorithm on the training set, which does not affect the limitations of our method.