分类号_____ 编 号_____

U D C_____ 密 级_____

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

本科生毕业设计（论文）

题　　目：基于自校正大型语言模型的算法自动设计方法研究

姓　　名：徐思创

学　　号：12011311

院　　系：计算机科学与工程系

专　　业：计算机科学与技术

指导教师：史玉回　讲席教授

2024 年　6　月　7 日

诚信承诺书

1.本人郑重承诺所呈交的毕业设计（论文），是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料均真实可靠。

2.除文中已经注明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确的方式标明。

3.本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。

4.在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

作者签名：徐思创

2024 年 6 月 7 日

基于自校正大型语言模型的算法自动设计方法研究

徐思创

（计算机科学与工程系　指导教师：史玉回）

[摘要] 最优化问题在各个领域都频繁出现，高效解决最优化问题具有重要意义。人工设计并使用元启发式算法，在处理复杂或非凸问题时，会遇到依赖专家经验、受专家偏好影响等问题。本研究对此问题提出了一种新的解决方案：利用自校正大语言模型自动设计元启发式算法。文章介绍了如何使用自校正大语言模型进行自动算法设计，其中详细介绍了提示词的设计与整体自动算法的设计，并阐述了研究方法和验证指标。实验结果表明，自校正大语言模型在自动设计元启发式算法方面具有显著优势，能够在更少的迭代中生成性能更优的算法。此外，实验使用消融分析的研究方法，证实了自校正模块对提高大语言模型性能的重要性。本研究完成了使用自校正大语言模型进行自动算法设计的目标。相比于直接使用大语言模型，这种新方法不仅提高了算法设计的效率和质量，还减少了人工成本，显示出强大的适应性和稳健性。这些成果对于推动最优化问题求解的自动化和智能化具有重要意义。

[关键词] 大语言模型、算法设计、自校正

[ABSTRACT] Optimization problems frequently arise in various fields, and efficiently solving them is of great significance. When manually designing and using metaheuristic algorithms to handle complex or non convex problems, one may encounter problems such as relying on expert experience and being influenced by expert preferences. This study proposes a new solution to this problem: using self correcting large language models to automatically design meta heuristic algorithms. The article introduces how to use a self correcting large language model for automatic algorithm design, including a detailed introduction to the design of prompt words and the overall automatic algorithm, as well as the research methods and validation indicators. The experimental results show that the self correcting large language model has significant advantages in automatically designing metaheuristic algorithms, and can generate algorithms with better performance in fewer iterations. In addition, the experimental use of ablation analysis confirmed the importance of the self correction module in improving the performance of large language models. This study achieved the goal of using a self correcting large language model for automatic algorithm design. Compared to directly using large language models, this new method not only improves the efficiency and quality of algorithm design, but also reduces labor costs, demonstrating strong adaptability and robustness. These achievements are of great significance for promoting the automation and intelligence of optimization problem solving.

[Keywords] large language model, algorithm design, self-correcting

目录

1. 引言

1.1 研究背景与意义

最优化问题，又称优化问题，是指从所有可行解中找到最优良的解的问题。具体来说，它通常在满足一系列约束条件时，最大化或最小化一个目标函数[1]。在数学、工程学、计算机科学等领域中，存在大量不同的最优化问题。快速、有效的解决这些最优化问题，可以提高经济与资源管理效率，促进数学与工程学科理论的发展，推动科学技术发展，具有重要的意义。

1.2 现有工作面临的问题与不足

目前，求解最优化问题主要有两个方向：一是通过数学求解器来进行求解，例如使用梯度下降，牛顿法和拟牛顿法这些方法。这些算法可以有效地解决一些简单的最优化问题，但在面对某些复杂的或非凸的最优化问题时，一般采用另一个方向：通过元启发式求解器进行求解。元启发式算法是一种基于经验和规则的算法，它不会通过完全遍历所有可能的解决方案来找到最佳解决方案，而会在可接受的计算时间和空间内给出最优化问题的一个可行的解决方案。[2] 常见的元启发式算法包括遗传算法，模拟退火算法，蚁群算法等。

目前，如果仅使用人工手动设计元启发式算法，有以下主要问题：

1.时间长、效率低：手动设计算法通常需要大量的时间和人力资源来进行研究、测试和调整；

2.创新性低：手动设计算法依赖于设计者的知识和经验，其可能会受到思维定势的限制；

3.泛化能力不足：手动设计的算法往往针对特定问题进行优化，可能在新问题上的表现不佳。

1.3 解决方案

为了更好地设计元启发式算法，并解决上文提到的设计元启发式算法遇到的问题，本研究计划使用大语言模型，来自动设计元启发式算法求解器。

大语言模型是一种人工智能模型，旨在理解和生成人类语言。大语言模型在大量的文本数据上进行训练后，可以像人类一样理解和生成文本以及其他形式的内容，并让它出现小语言模型没有的优势[3]：

1.上下文学习：大语言模型不需要额外的训练或梯度更新，就可以根据提供的自然语言指令或任务，生成预期的结果；

2.指令遵循：大语言模型能够在不使用显式样本的情况下，通过理解任务指令来执行新任务，泛化能力得到了提高；

3.循序渐进的推理：大语言模型可以通过包括推理中间步骤的方法，解决涉及多个推理步骤的复杂任务。

具有这些优势的大语言模型，可以执行广泛的任务，同样可以在没有预定义设计空间的情况下生成算法。且其作为易用的人机交互代理，在自动设计元启发式算法求解器的方向有很大作用。因此，基于大语言模型进行自动算法设计的可行性较高。

1.4 主要贡献

根据现有研究，使用大语言模型来自动设计元启发式算法有以下主要贡献[4]：

1.开放式启发式空间：与传统的启发式算法相比，大语言模型提供了一个更广阔的启发式搜索空间，使得到的算法不受预定义的启发式方法的限制；

2.改进算法性能：大语言模型辅助设计的启发式算法，在某些情况下能够胜过人工设计的算法，展示了大语言模型在复杂和新颖的实际应用中自动化算法设计的潜力；

3.减少人工成本：使用大语言模型自动设计算法，减少了设计者在算法设计过程中的参与，节省了时间与人工资源。

1.5 论文结构

本文分为以下几个部分：

1.引言部分首先介绍了最优化问题的重要性及其应用领域，通过目前解决最优化问题方法的不足，引出了本文的解决方法：使用大语言模型设计算法的方法；

2.基础理论部分讨论了大语言模型存在的问题，为了解决问题，参考自校正大语言模型的理论，提出了对大语言模型进行自我评估的自校正方法；

3.项目设计部分详细讨论了项目的提示词与自动算法是如何设计的，并阐明了使用的消融分析的研究方法；

4.结果部分先分别展示了改进提示词，使用自校正方法对结果的改进与优化，

然后展示了自校正大语言模型在解决数个最优化问题时的结果；

5.总结部分总结了本研究的成果的价值，意义与特色，并强调了自校正大语言模型在自动算法设计中的优势。

2. 基础理论与相关工作

2.1 理论知识

大语言模型具有优秀的性能，但在使用大语言模型时，得到的结果也存在以下几个主要问题：

1.幻觉：大语言模型可能会编造事实或引用不存在的来源；

2.不忠实的推理：大语言模型得出的结论可能不符合先前生成的推理链；

3.有毒、有偏见和有害内容：由于训练数据中存在不足，大语言模型可能会生成有毒、有偏见或有害的内容；

4.有缺陷的代码：大语言模型生成的代码可能存在语法错误或格式错误等问题。

因此，在使用大语言模型时，为了提高准确性和效率，需要对大语言模型进行自校正。大语言模型的自校正是指模型在生成输出后，自动评估和修正其输出中的错误或不准确的信息。以在没有外部反馈的情况下，提高模型输出的质量和可靠性。

2.2 参考的研究方法

为了实现大语言模型的自校正，本研究首先参考了 Michael Saxon 等研究者对自校正大语言模型的研究[5]。他们尝试并分析了多种大语言模型进行自校正的策略，并对这些策略进行了分类，同时总结了自校正策略的主要应用，并讨论了未来的发展方向和挑战。这项工作使大语言模型更加实用和可部署，同时减少了大语言模型对人类反馈的依赖。

在他们的研究中，根据反馈的阶段，这些策略主要可以分为训练时校正，生成时校正与事后校正。为了确定综合效果更好的自校正方法，本研究同时参考了 Hong Sun 等研究者对大语言模型自动提示工程的研究[6]。他们按照类似于大语言模型自校正策略中自我评估的方法，先提供一个初始提示词，并指导大语言模型为选定样本推导出新的提示词，然后从每个样本的提示词中总结，并将结果添加回初始提示词，以形成新的、丰富的提示词。他们同时在 BIG-Bench Instruction Induction 数据集上进行了评估，得到的结果表明，该方法能够显著提高多个任务的准确性。

2.3 选用的研究方法

根据上文所述，为了保证自校正的效果，本研究选择的大语言模型自校正方法为：在大语言模型生成时进行校正，并使用自我评估的方法进行校正。

3. 项目设计

3.1 项目设计

3.1.1 提示词设计

使用大语言模型进行算法设计，需要合理、有效的提示词来引导大语言模型生成符合预期的结果。在编写提示词时，应该尽量遵从以下原则：

1.提供清晰和具体的指令：

对输入的语句，可以使用分隔符清楚的指示输入的不同部分，来避免输入的文本可能存在一些误导性的话语对模型造成干扰。同时，可以要求模型结构化的输出，例如要求模型输出 json 或其他格式。

2.给模型时间来思考：

可以指定模型完成任务所需的步骤，让模型一步一步来解，使模型在输出 token 的时候可以参照上一个步骤的结果，从而提升输出的正确率。

根据以上原则，分别在自动算法中调用大语言模型的不同时间，设计当次对话所需的提示词。

在首次与大语言模型进行对话，要求其生成最优化问题的元启发式算法时，提示词分为以下几个部分：

1.首先向大语言模型明确对话目的：设计元启发式算法，然后告诉大语言模型接下来的部分会有哪些分隔符，每一对分隔符中包含的具体是问题的哪一部分；

2.在接下来的每一对分隔符中，分别向大语言模型描述问题，描述评估标准，描述评估用的方法及代码，描述可以解决这个问题的样例算法及代码；

3.最后向大语言模型提出生成算法时的要求与限制，并规范生成的结果的格式。

具体的提示词示例如下：

Assuming you are an expert in the field of evolutionary computation and metaheuristic algorithm design. You need to modify the algorithm based on the example to make it perform better.

The optimization problem description is delimited by '<problem description> </problem description>', which details the source and mathematical expression of the target problem.

Metrics delimited by '<metrics> </metrics>', which indicates the evaluation criteria of the algorithm's performance in solving the target problem.

The existing code is delimited by '<existing code> </existing code>', which gives the existing

---

code that does the initialization and other tasks, you need to read it carefully, but you cannot change it.

Examples of the metaheuristic algorithms are delimited by '<examples> </examples>', which provides some examples of metaheuristic algorithms we have designed to help you understand the requirements of algorithm design and design algorithms that perform better than the examples.

<problem description>

The description of the problem

</problem description>

<metrics> fitness value: the value of the function equation f(x). Smaller fitness value indicates better algorithm performance. </metrics>

<existing code>

```
evaluate code
```

</existing code>

<examples>

<example>

#metrics: "avg_fitness": avg_fitness

```
example code
```

</example>

</examples>

Please imitate the example I provided and output your algorithm results. The example uses a genetic algorithm, so you must make some modifications based on the example to make your algorithm perform better than the example.

Modification includes modifying the probability of crossover or mutation occurring in each generation. Do not modify the values of population_size and generation_number in the algorithm. Ensure that the result you output have made certain changes compared to the example. In your result, the value of population_size must be 100 and the value of generation_number must be 200.

The result you output should be the algorithm I requested, in the format of starting with 'def SOLVE(toolbox, lower_bound, upper_bound):' and end with 'best_ind = tools.selBest(pop, 1)[0] \n return best_ind.fitness.values[0]'.

After outputting your algorithm, tell me what modifications have been made to your algorithm based on example.

在得到大语言模型生成的算法并评估后，需要再次与大语言模型进行对话，

并要求其给出改进建议，以实现大语言模型的自校正，此时的提示词分为以下几个部分：

1. 首先向大语言模型明确对话目的：对比样例元启发式算法与大语言模型生成的元启发式算法，然后告诉大语言模型接下来的部分会有哪些分隔符，每一对分隔符中包含的具体是问题的哪一部分；

2. 在接下来的每一对分隔符中，分别向大语言模型描述问题，描述评估标准，描述评估用的方法及代码，描述上文所说的两个算法，算法的表现及代码；

3. 最后向大语言模型询问两个算法的不同，它们表现的差异，以及对生成的算法下一步的改进建议，并规范生成的结果的格式。

具体的提示词示例如下：

Assuming you are an expert in the field of evolutionary computation and metaheuristic algorithm design. Your need to design a metaheuristic algorithm to solve an optimization problem.

For the following problem, you need to compare the differences between the two algorithms used to solve the problem and provide improvement suggestions based on their performance differences.

The optimization problem description is delimited by '<problem description> </problem description>', which details the source and mathematical expression of the target problem.

Metrics delimited by '<metrics> </metrics>', which indicates the evaluation criteria of the algorithm's performance in solving the target problem.

The existing code is delimited by '<existing code> </existing code>', which gives the existing code that does the initialization and other tasks, you need to read it carefully, but you cannot change it.

Examples of the metaheuristic algorithms are delimited by '<examples> </examples>', which provides some examples of metaheuristic algorithms we have designed to help you understand the requirements of algorithm design and design algorithms that perform better than the examples.

<problem description>

The description of the problem

</problem description>

<metrics>

fitness value: the value of the function equation f(x). Smaller fitness value indicates better algorithm performance.

</metrics>

<existing code>

evaluate code

</existing code>

<examples>

<example1>

#metrics: "avg_fitness": avg_fitness

example code

</example1>

<example2>

#metrics: "avg_fitness": avg_fitness

generated code

</example2>

</examples>

Please write down the differences between these two algorithms. I want to improve the performance of the second algorithm, so provide some suggestions for improvement based on the second algorithm.

In your output, first output the differences between the two algorithms, then output "My advice:", and finally output one short but specific suggestion. Do not modify the values of population_size and generation_number in your suggestion.

在得到改进建议后，在迭代与大语言模型进行对话时，提示词与首次与大语言模型对话时的提示词相似，在其基础上添加了这条改进建议，并表明这是大语言模型自己生成的建议，在设计算法时需要尽量参考这条建议。

3.1.2 自动算法设计

使用自校正大语言模型来生成元启发式算法，需要与大语言模型进行多轮对话，以迭代的方式生成最优的结果。在迭代中，需要评估每次生成的结果，选取其中更好的结果，加入提示词后重新与大语言模型进行对话，从而使大语言模型可以通过多次生成结果来改进自己的结果。由于大语言模型拥有记忆与语义合成能力，它可以逐渐优化算法代码，从而得到更好的结果。

基于以上目标，本研究设计的自动算法设计流程分为以下几个步骤：

1. 对于每一个需要解决的最优化问题，首先提供一个可以解决这个问题的样

例算法，并将其加入第一次与大语言模型对话的提示词，与模型进行对话，要求其参考例子，生成自己的比样例更好的元启发式算法。

2.使用上文设计的提示词，得到指定格式的模型输出后，从其中分离出生成的算法代码，并对生成的算法进行评估。其中，评估的方法为，将生成的算法代码与评估代码结合，直接运行算法代码，得到的结果直接代入最优化问题的目标函数，将目标函数的值作为适应度分数。若该最优化问题为最小化问题，则适应度分数的值越小，算法效果越好；若问题为最大化问题，则适应度分数的值越大，算法效果越好。

3.保存模型生成的算法及其适应度分数到所有生成的算法集后，将样例算法与其适应度分数，模型生成的算法与其适应度分数加入自校正的提示词，与大语言模型再次进行对话，得到模型对算法的改进建议。

4.对于接下来的每一轮对话，选择目前适应度分数最好的算法作为样例算法，再次要求其生成自己的比样例更好的元启发式算法，并在提示词中加入上一轮最后得到的改进建议。

具体流程图如图 3.3：

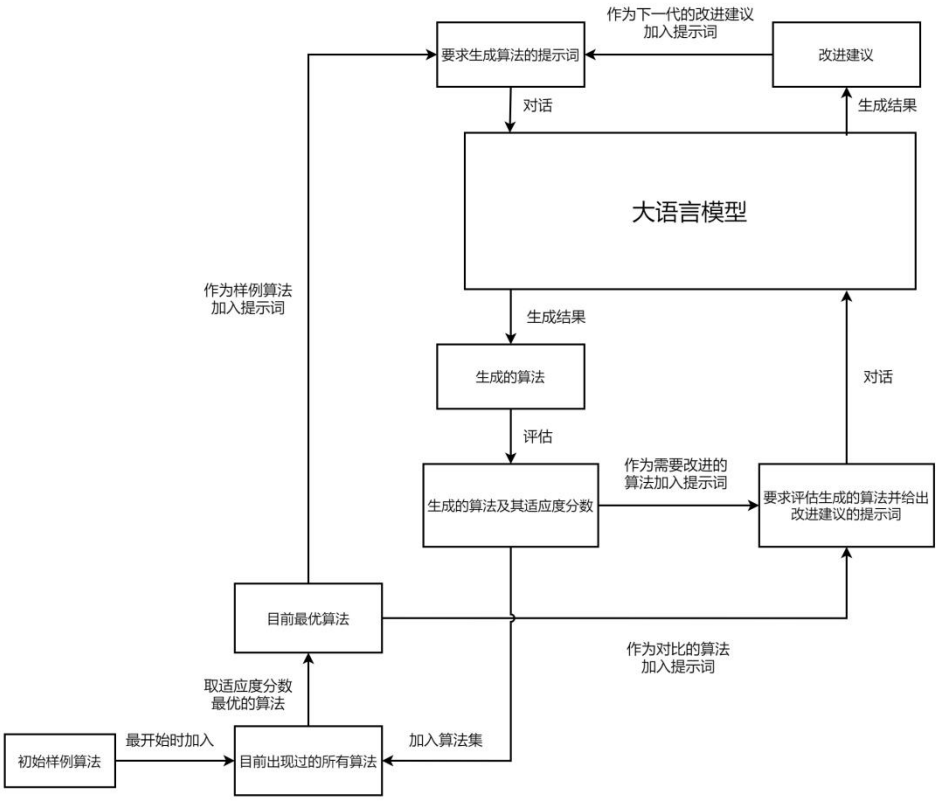

图 3.3：使用自校正大语言模型进行自动算法设计的流程图

3.2 验证指标

在大语言模型迭代生成元启发式算法时，我们保存每一代生成的算法，以及算法在评估后得到的适应度分数。首先，可以对比生成的算法与样例的适应度分数，判断生成的算法是否效果更好。其次，可以对比每一代生成的算法与上一代的区别，判断大语言模型是否不断尝试优化，改进算法。

4. 实验结果

本研究使用 Python 编写代码，使用 CodeLlama-7B-Instruct 大语言模型，这个模型基于开源的 Llama 2 模型开发。Llama 2 是由 Meta AI 在 2023 年发布的一系列预训练和微调的大型语言模型，能够执行从文本生成到编程代码的各种自然语言处理任务[8]。而 CodeLlama-7B-Instruct 基于 Llama 2 模型权重初始化，并在一个重点代码的数据集上进行了大量训练，使其能够处理长文本上下文，并且经过指令优化，使其在执行编程任务时更安全。

用于测试的最优化问题来源于 CEC2005 测试集[9]，这个测试集来源于 IEEE 国际进化计算大会，这是进化计算领域中规模最大、影响最重要的会议之一。CEC2005 测试集是用于评估和比较优化算法性能的标准函数集合，这些函数通常具有已知的最优解或者近似最优解，可以用来测试优化算法的搜索能力、收敛速度和准确性。由于最大化与最小化问题可以进行转换，测试集中的所有测试函数都是最小化问题。

4.1 提示词优化结果

本研究为了让大语言模型正确的理解提示词的要求，并生成提示词期望的结果，按照前文所述的设计提示词需要遵从的原则，多次修改、精简了提示词。图 4.1.2、图 4.1.3 展示了优化提示词前后，迭代中每一代的最优算法的适应度分数的变化趋势：

This is the f2 problem from the CEC_2005 competition.
It's a minimization problem with the mathematical expression $f(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i} z_j\right)^2+f\_bias_1(z=x-o,x=[x_1,x_2,...,x_D],o=[o_1,o_2,...,o_D])$, where x is the decision variable, D is the dimension of x and o is the shifted global optimum. The value of D is set to 30 concretely. The range of the value of x is $\[-100,100\]$.

图 4.1.1：验证提示词优化结果时使用的最优化问题

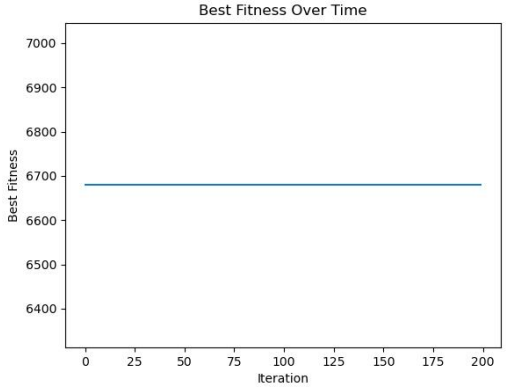

图 4.1.2：优化提示词前每一代保存的算法集中最好的算法的适应度分数

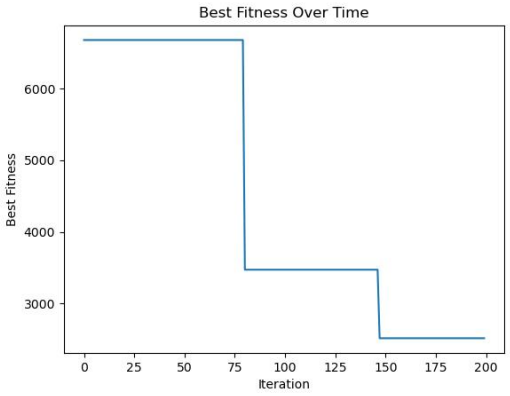

图 4.1.3：优化提示词后每一代保存的算法集中最好的算法的适应度分数

图中横轴为设计算法时大语言模型的迭代代数，纵轴为适应度分数，折线为每一代保存的算法集中，最好的算法的适应度分数。研究初期设计的提示词由于内容冗余，要求不明确，导致大语言模型只能遵守提示词中格式要求，却错误地复制样例算法将其作为自己生成的算法。因此，大语言模型生成的算法或是错误的修改导致算法恶化，或是并没有自己改进、优化算法，导致保存的算法集的最优适应度分数没有改变，适应度分数呈现为一条横线。而在多次修改、精简提示词后，大语言模型可以自行设计并优化算法，得到了适应度分数更低，效果更好的算法。

4.2 消融分析自校正模块结果

为了验证自校正的大语言模型相比不进行校正的大语言模型的优势，本研究采用了消融分析的方法，在移除自校正模块后观察对生成的算法结果的影响。本研究同样设计了使用不进行自校正的大语言模型的自动设计算法的方法，具体来说，相比自校正的大语言模型，这个模型不会进行自我评估，因此每一轮迭代只进行一次对话，且每一轮更改提示词时，只提供生成时结果更好的算法，而不提供改进建议。

本研究对于同一个最优化问题，分别在直接使用大语言模型与使用自校正大语言模型的情况下，进行自动算法设计。使用的最优化问题如图 4.2.1：

This is the f2 problem from the CEC_2005 competition.
It's a minimization problem with the mathematical expression $f(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i} z_j\right)^2+f\_bias, (z=x-o, x=[x_1,x_2,...,x_D], o=[o_1,o_2,...,o_D])$, where x is the decision variable, D is the dimension of x and o is the shifted global optimum. The value of D is set to 30 concretely. The range of the value of x is \[$-100, 100$ \].

图 4.2.1：验证大语言模型自校正模块作用时使用的最优化问题

首先对比两种方法最终得到的算法的适应度分数，如图 4.2.2、图 4.2.3：

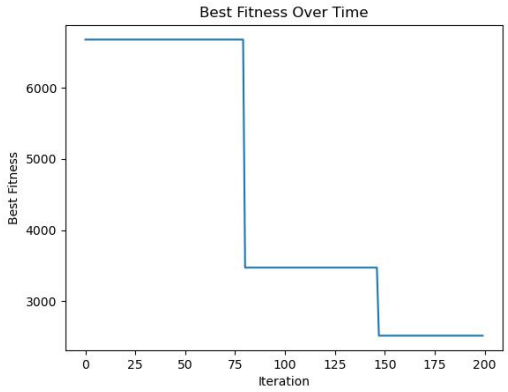

图 4.2.2：使用非自校正大语言模型每一代保存的算法集中最好的算法的适应度分数

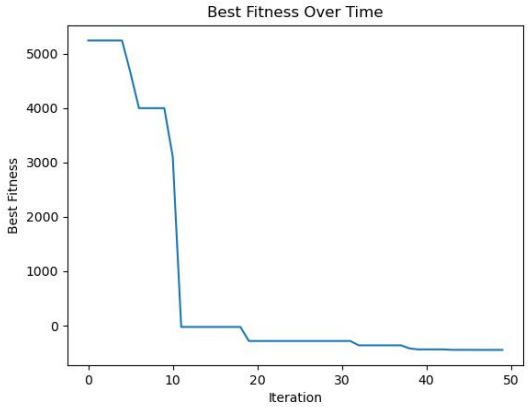

图 4.2.3：使用自校正大语言模型每一代保存的算法集中最好的算法的适应度分数

图中横轴为设计算法时大语言模型的迭代代数，纵轴为适应度分数，折线为每一代保存的算法集中，最好的算法的适应度分数。从图中可以看出，在加入了自校正的方法后，大语言模型使用更少的迭代代数，得到了适应度分数更低，表现更好的算法。

然后对比两种方法每一代得到的算法及其适应度分数，分析每一代生成的算法与上一代算法与适应度分数的区别，判断大语言模型是否不断尝试优化，改进算法，改进算法后是否使适应度分数降低，得到的结果如图 4.2.4、图 4.2.5：

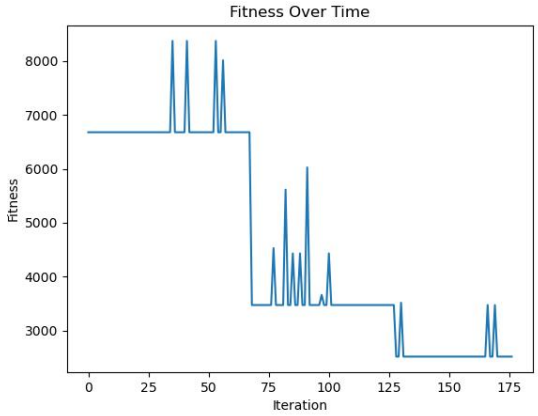

图 4.2.4：使用非自校正大语言模型每一代生成的算法的适应度分数

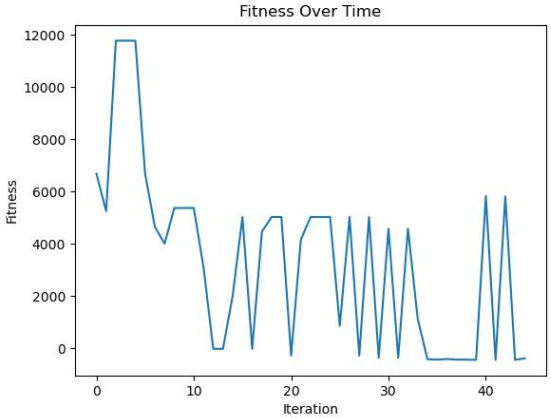

图 4.2.5：使用自校正大语言模型每一代生成的算法的适应度分数

图中横轴为设计算法时大语言模型的迭代代数，纵轴为适应度分数，折线为每一代生成的算法的适应度分数。从图中可以看出，在加入了自校正的方法后，大语言模型修改，优化算法的频率明显更高，且进行的有效修改，即导致适应度分数降低的修改更多，说明自校正大语言模型对算法的优化效果更好。

4.3 系统最终结果

在验证了自校正大语言模型在自动算法设计的能力后，本研究同样尝试了 CEC2005 测试集中的其他最优化问题，使用自校正大语言模型进行自动算法设计，得到了以下结果：

1. Shifted Rosenbrock’s Function

This is the f6 problem from the CEC_2005 competition.
It's a minimization problem with the mathematical expression

$f(x)=\sum_{i=1}^{D-1}(100(z_i^2-z_{i+1})^2+(z_i-1)^2)+f\_bias,(z=x-o+1,x=[x_1,x_2,...,x_D],o=[o_1,o_2,...,o_D])$, where x is the decision variable, D is the dimension of x and o is the shifted global optimum. The value of D is set to 30 concretely. The range of the value of x is $\[-100,100\]$.

图 4.3.1：Shifted Rosenbrock’s Function 对应的具体最优化问题

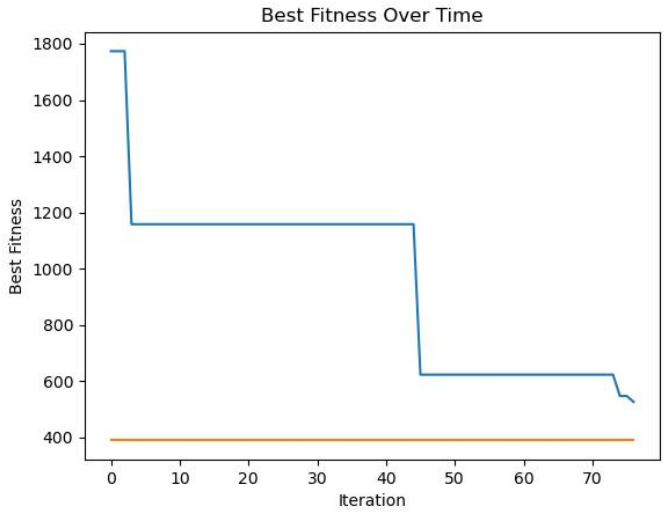

图 4.3.2：使用自校正大语言模型每一代保存的算法集中最好的算法的适应度分数

2. Shifted Rotated High Conditioned Elliptic Function

This is the f4 problem from the CEC_2005 competition.
It's a minimization problem with the mathematical expression $f(x)=\sum_{i=1}^{D}(10^6)^{\frac{i-1}{D-1}}z_i^2+f\_bias_4(z=x-o,x=[x_1,x_2,...,x_D],o=[o_1,o_2,...,o_D])$, where x is the decision variable, D is the dimension of x and o is the shifted global optimum. The value of D is set to 30 concretely. The range of the value of x is $\[-100,100\]$.

图 4.3.3:Shifted Rotated High Conditioned Elliptic Function 对应的具体最优化问题

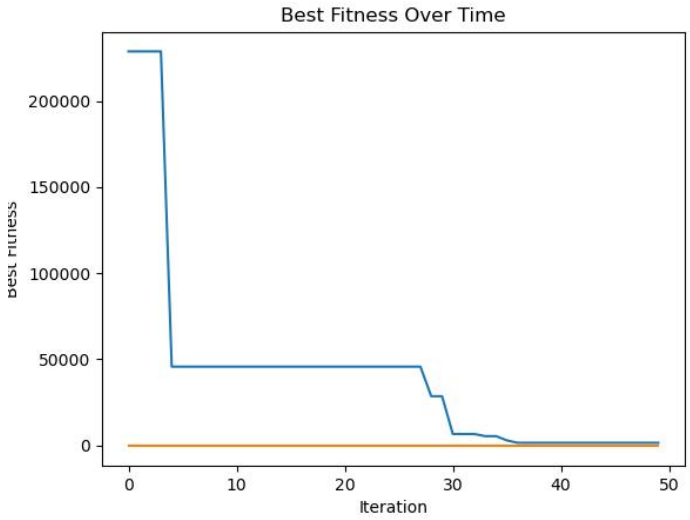

图 4.3.4：使用自校正大语言模型每一代保存的算法集中最好的算法的适应度分数

3. Shifted Sphere Function

This is the f1 problem from the CEC_2005 competition.
It's a minimization problem with the mathematical expression $f(x)=\sum_{i=1}^{D}z_i^2+f\_bias_1(z=x-o,x=[x_1,x_2,...,x_D],o=[o_1,o_2,...,o_D])$, where x is the decision variable, D is the dimension of x and o is the shifted global optimum. The value of D is set to 30 concretely. The range of the value of x is $\[-100,100\]$.

图 4.3.5:Shifted Sphere Function 对应的具体最优化问题

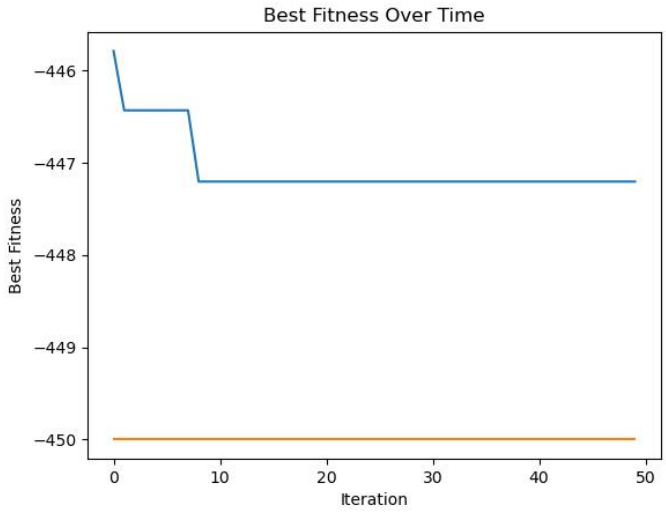

图 4.3.6：使用自校正大语言模型每一代保存的算法集中最好的算法的适应度分数

4. multi-modal function

This is the f3 problem from the CEC_2005 competition.
It's a minimization problem with the mathematical expression $f(x)=\sum_{i=1}^{D}|z_i|+\prod_{i=1}^{D}|z_i|+f\_bias, (z=x-o, x=[x_1,x_2,...,x_D], o=[o_1,o_2,...,o_D])$, where x is the decision variable, D is the dimension of x and o is the shifted global optimum. The value of D is set to 30 concretely. The range of the value of x is $\[-10, 10\]$.

图 4.3.7:multi-modal function 对应的具体最优化问题

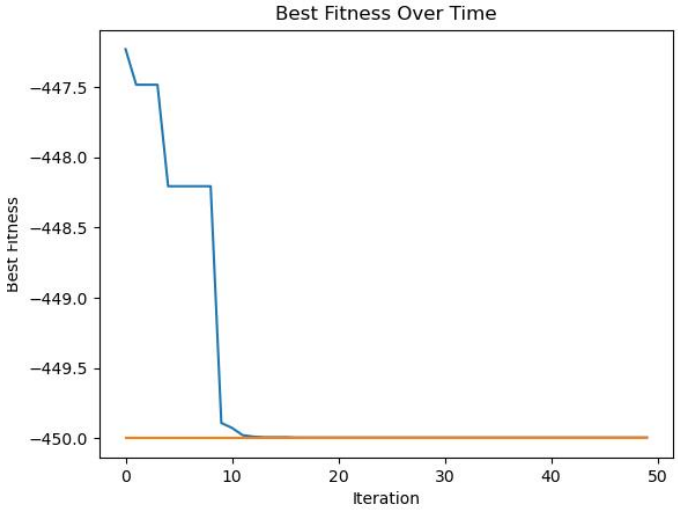

图 4.3.8：使用自校正大语言模型每一代保存的算法集中最好的算法的适应度分数

5. Shifted Schwefel’s Problem 1.2

This is the f2 problem from the CEC_2005 competition.
It's a minimization problem with the mathematical expression $f(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}z_j\right)^2+f\_bias, (z=x-o, x=[x_1,x_2,...,x_D], o=[o_1,o_2,...,o_D])$, where x is the decision variable, D is the dimension of x and o is the shifted global optimum. The value of D is set to 30 concretely. The range of the value of x is $\[-100, 100\]$.

图 4.3.9:Shifted Schwefel’s Problem 1.2 对应的具体最优化问题

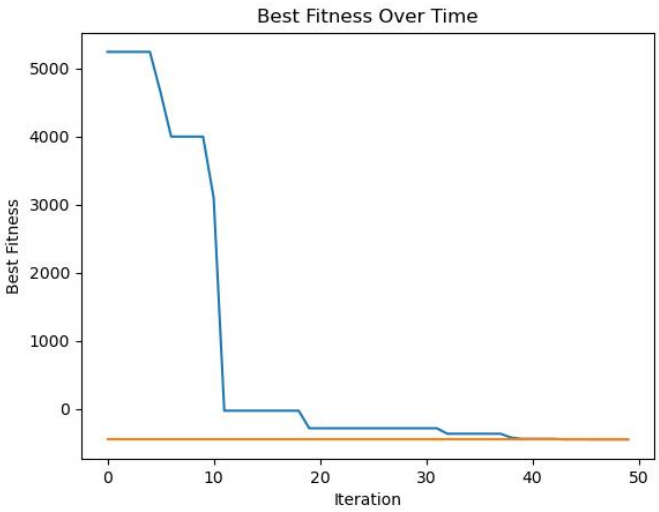

图 4.3.10：使用自校正大语言模型每一代保存的算法集中最好的算法的适应度分数

图中横轴为设计算法时大语言模型的迭代代数，纵轴为适应度分数，蓝线为每一代保存的算法集中，最好的算法的适应度分数，红线为该最优化问题的最优解。在不同的问题与样例中，自校正大语言模型都能改进，优化算法，使算法的适应度分数降低，并趋向于接近最优解，说明使用自校正大语言模型进行自动算法设计的方法在不同的问题上都有效，方法具有稳健性与可靠性。

5. 总结

本研究成功完成了使用自校正大语言模型进行自动算法设计的任务。这对帮助求解最优化问题有很大的意义，在自校正大语言模型的辅助下，设计最优化问题的元启发式算法更加方便、快捷。

相比只使用普通的大语言模型，使用自校正大语言模型有以下优点与特色：

1.自我纠正能力：自校正大语言模型能够更快的改进算法，提高了生成算法的质量；

2.减少人工干预：通过自校正机制，大语言模型减少了对人类标注数据的依赖，降低了成本和时间消耗；

3.适应性强：自校正大语言模型能够更好的适应新的数据集和任务，而不需要从头开始训练或额外数据集进行微调。

在本研究中，创造性成果为使用了自校正的大语言模型。自校正大语言模型能够生成更加准确和可靠的算法，这对于提高自动化系统的性能至关重要。同时，自校正模型在生成代码或文本时，能够通过自我评估减少错误和偏差，提高了生成结果的可信度和实用性。

致谢

在本研究完成之际，我要特别感谢我的导师史玉回老师和赵琪老师。在整个研究过程中，两位老师每周都要抽时间对我的研究进行指导与答疑，帮助我克服了研究中的种种困难。同时，我要感谢我的学校南方科技大学提供的优良学习和研究环境。感谢学校提供的各种资源和机会，让我能够顺利完成我的学业和研究。

参考文献

[1] Boyd, S. P.; Vandenberghe, L. Convex Optimization[M]. Cambridge: Cambridge University Press, 2004: 129.

[2] Myers, D. G. Social psychology[M]. Tenth. New York, NY: McGraw-Hill, 2010: 94.

[3] Zhao, W. X.; Zhou, K.; Li, J. et al. A Survey of Large Language Models[EB/OL]. arXiv:2303.18223, 2023.

[4] Ye, H.; Wang, J.; Cao, Z.; Song, G. ReEvo: Large Language Models as Hyper-Heuristics with Reflective Evolution[EB/OL]. arXiv:2402.01145, 2024.

[5] Pan, L.; Saxon, M.; Xu, W.; Nathani, D.; Wang, X.; Wang, W. Y. Automatically Correcting Large Language Models: Surveying the landscape of diverse self-correction strategies[EB/OL]. arXiv:2308.03188, 2023.

[6] Sun, H.; Li, X.; Xu, Y.; Homma, Y.; Cao, Q.; Wu, M.; Jiao, J.; Charles, D. AutoHint: Automatic Prompt Optimization with Hint Generation[EB/OL]. arXiv:2307.07415, 2023.

[7] Sahoo, P.; Singh, A. K.; Saha, S.; Jain, V.; Mondal, S.; Chadha, A. A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications[EB/OL]. arXiv:2402.07927, 2024.

[8] Touvron, H.; Martin, L.; Stone, K. et al. Llama 2: Open Foundation and Fine-Tuned Chat Models[EB/OL]. arXiv:2307.09288, 2023.

[9] Suganthan, P. N.; Hansen, N.; Liang, J. J.; Deb, K.; Chen, Y.-P.; Auger, A.; Tiwari, S. Problem Definitions and Evaluation Criteria for the CEC 2005 Special Session on Real-Parameter Optimization[R]. Nanyang Technological University, Singapore, May 2005 and KanGAL Report #2005005, IIT Kanpur, India.

[10] Chen, A.; Dohan, D. M.; So, D. R. EvoPrompting: Language Models for Code-Level Neural Architecture Search[EB/OL]. arXiv:2302.14838, 2023.

[11] Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; Chen, W. LoRA: Low-Rank Adaptation of Large Language Models[EB/OL]. arXiv:2106.09685, 2021.

[12] Gigerenzer, G.; Todd, P. M. et al. Simple heuristics that make us smart[M]. New York: Oxford University Press, 1999.

[13] Pillkahn, U. Innovationen zwischen Planung und Zufall: Bausteine einer Theorie der

bewussten Irritation[M]. Books on Demand, 2012: 170.

[14] Ausiello, G. et al. Complexity and Approximation Corrected[M]. Springer, 2003.

[15] Chen, K.; Yang, Y.; Chen, B.; Hernández López, J. A.; Mussbacher, G.; Varró, D. Automated Domain Modeling with Large Language Models[A]. In Proceedings of the MODELS 2023[C]. IEEE, 2023: 162-172.

[16] Chen, B.; Yi, F.; Varró, D. Prompting or Fine-tuning? A Comparative Study of Large Language Models for Taxonomy Construction[A]. In MDE Intelligence[C], 2023.

[17] Poli, R.; Langdon, W. B.; McPhee, N. F. A Field Guide to Genetic Programming[M]. Lulu.com, 2008.

[18] Banar, B.; Colton, S. Autoregressive Self-Evaluation: A Case Study of Music Generation Using Large Language Models[A]. In Proceedings of the International Conference on Computational Creativity[C], 2023: 264-272.

[19] Wang, Y. On Finetuning Large Language Models[J]. Political Analysis, 2023.

[20] Sabbatella, A.; Ponti, A.; Giordani, I.; Candelieri, A.; Archetti, F. Prompt Optimization in Large Language Models[J]. Mathematics, 2024, 12: 929.