# GP retrosynthesis - 项目问题分析与改正

下面这段就是给“自动改代码大模型”的提示词，你可以直接整段复制过去用（如果有 token 限制，可以稍微裁剪注释，但保留关键点）：

---

你是一个擅长化学信息学和遗传规划（Genetic Programming, GP）的高级代码修复助手。
我有一个已经搭建好的项目，目标是：**用遗传搜索/遗传规划做分子逆合成**。

项目大致结构（仅供参考，实际以你扫描仓库为准）：

- 入口脚本：`run_gp_search_demo.py`（玩具/示例 GP 逆合成）
- 核心模块（名称可能略有出入）：
  - 表示层：`gp_retro_repr/`（Program、Route、ReactionTemplate 等）
  - 可行性层：`gp_retro_feas/`（FeasibilityEngine, FeasibleExecutor, action mask 等）
  - 目标/适应度层：`gp_retro_obj/`（RouteFitnessEvaluator, audit 函数等）
  - 通用 GP 框架：`gp_core/`（program_ops、search、selection、mutation 等）
- 模型与数据：`scscore/`、`data/`（building blocks, reaction templates, target smiles 等）
- 仓库里还有一些第三方参考工程：`synga-FE64/`, `ChemProjector-main/`, `LLM-Syn-Planner-main/`，它们主要是参考，不是当前 GP 主流程的核心依赖。

现在的问题是：**在运行 GP 搜索的时候，几乎找不到任何目标分子的可合成/逆合成路径，基本上 `solved=0`，看不到一条完整的逆合成路径。**
我怀疑问题主要集中在以下几个方面，请你系统检查并改好代码，使得搜索至少能在简单例子上找到 1 条成功的逆合成路径。

> 重要：请优先关注 *我的 GP 实现逻辑是否自洽* 和 *针对逆合成任务的抽象是否正确落实到代码*，而不要对整个架构做颠覆性重构。

---

## 需要你重点检查和修复的内容

### 1. 反应模板应用（`ReactionTemplate.apply_to_product`）是否真正工作

- 在 `gp_retro_repr`（或类似）的模块中找到负责将 SMIRKS/模板应用到产物分子、生成反应物集合的函数，通常叫：
  `ReactionTemplate.apply_to_product(product_smiles: str)`。
- 该函数应优先使用 `rdchiral`（`rdchiralReaction`, `rdchiralReactants`, `rdchiralRun`），在 `rdchiral` 不可用时可以用 RDKit (`rdChemReactions.ReactionFromSmarts`) 做兜底。
- 修复点：
  1. **不能静默失败**：如果 `rdchiral` 导入失败或模板应用抛异常，不能只返回空列表而不记录原因。
  2. 请增加合理的异常处理和日志（或向上抛异常），以便上层能知道是“模板没打上”还是“化学引擎没装好”。

3. 将输出反应物集合做 SMILES 规范化和去重（比如用 **`MolToSmiles(..., canonical=True, isomericSmiles=True)`**）。

## 2. Program 的语法：每个 `ApplyTemplate` 前必须有 `Select`

- 在项目中找到 Program 结构与执行逻辑（一般在 **`gp_retro_repr/program.py`** 或类似文件），它会按顺序执行指令：
  - **`Select(i)`**：选择当前分子集合中的第 i 个分子；
  - **`ApplyTemplate(template_id)`**：对刚刚选中的分子应用模板；
  - **`Stop()`**：终止程序。
- 再在 **`run_gp_search_demo.py`**（以及任何 GP 相关脚本）中，找到 **“模板列表 → Program”** 的构造函数，例如：

```
python

def program_from_templates(template_ids):
    steps = [Select(0)]
    for tid in template_ids:
        steps.append(ApplyTemplate(tid))
    steps.append(Stop())
    return Program(steps)
```

- 这里常见的 bug：只在最前面加了一个 **`Select(0)`**，而 Program 执行逻辑通常会在一次 **`ApplyTemplate`** 后清空“当前选中分子”，如果下一条指令仍然是 **`ApplyTemplate`** 就会抛出类似 “ApplyTemplate encountered before Select” 的异常。
- 修复要求：
  1. **修改所有构造 Program 的地方**，保证对每一个 **`ApplyTemplate`** 前都显式插入一个 **`Select`**，例如：

```
python

def program_from_templates(template_ids):
    steps = []
    for tid in template_ids:
        steps.append(Select(0))                    # 每步都先 Select
        steps.append(ApplyTemplate(tid, rational="gp"))
    steps.append(Stop())
    return Program(steps)
```

  2. 检查 **`mutate_program`**, **`crossover_one_point`**, **`random_program`** 等 GP 操作，**保证不会生成语法非法的 Program**：
     - **`Select`** 与 **`ApplyTemplate`** 必须成对出现；
     - Program 最后必须以 **`Stop`** 结尾；
     - 不允许出现连续的 **`ApplyTemplate`** 而中间没有 **`Select`**；
     - 所有 **`Select`** 的 index 必须在当前分子集合范围内（至少在静态结构上不明显违规）。

## 3. 个体评估函数中不能吞掉关键异常

- 在 **`run_gp_search_demo.py`**（或其他入口脚本）中，找到评估个体的函数，例如 **`evaluate_program`** 或类似名字。

- 当前实现很可能类似：

```
python

try:
    route = executor.execute(program, target_smiles=target)
except Exception:
    route = DummyRoute(target)  # 或者用 Program([Stop()])
fitness = evaluator.evaluate(route)
```

- 这会把所有关键错误（例如化学引擎没装、Program 语法错误、模板匹配异常、库存/口罩问题）全部吞掉，使调试变得非常困难。
- 修复要求：
  1. 仍然可以保留容错（避免单个个体中断整个 GP 循环），但必须：
     - 记录 / 打印异常信息（包括异常类型、message、部分 Program 内容）；
     - 或在 debug 模式下直接向上抛异常。
  2. 推荐结构（伪代码）：

```
python

def evaluate_program(prog, exe, evaluator, target):
    try:
        route = exe.execute(prog, target_smiles=target)
    except Exception as e:
        log_or_print(f"[EVAL] execute(prog) failed: {repr(e)} | prog={prog}")
        try:
            route = exe.execute(Program([Stop()]), target_smiles=target)
        except Exception as e2:
            log_or_print(f"[EVAL] execute(Stop) also failed: {repr(e2)}")
            route = DummyRoute(target)
    fit = evaluator.evaluate(route)
    return {"program": prog, "route": route, "fitness": fit}
```

## 4. FeasibleExecutor 中的分子集合更新逻辑要正确、规范化

- 在 **`gp_retro_feas`** 中找到执行 Program 的核心逻辑，例如 **`FeasibleExecutor.execute`** 或相关函数。
- 在 **`ApplyTemplate`** 成功时，分子集合应该更新为：
  - 从旧集合中移除被选中的产物分子；
  - 把选中的反应物集合加入集合；
  - 对新的分子集合做规范化（canonical SMILES）和去重。
- 修复要求：
  1. 检查是否有错误的合并逻辑（例如把产物又加回去了，或者没有移除被选中分子）。
  2. 给更新前后加必要的断言或日志，确认：**`len(new_molecules)`** 合理；**`selected_index`** 不越界。
  3. 增加一个 **`canonicalize_unique`** 辅助函数，对分子集合做唯一化并与库存统一 SMILES 规范。

## 5. Action Mask（模板口罩）过于严格时的退化策略和调试信息

- 在 **`gp_retro_feas`** 中找到与“模板口罩/可用模板集合”相关的逻辑（例如 **`ActionMaskBuilder`**、**`build_action_mask`** 等）。

- 当前逻辑可能会因为功能团匹配/模板过滤过于严格，导致某些状态下可选模板集合为空，直接导致该 Program 失败。
- 修复要求：
  1. 增加 debug 日志：对于每个状态，记录
     - 当前产物 SMILES；
     - mask 中模板数量；
     - 若 mask 为空，记录过滤原因。
  2. 加一个合理的退化策略（开发/调试模式下即可）：
     - 若 mask 非空 → 使用 mask；
     - 若 mask 为空 → 回退为 “所有模板中的 Top-K 子集”（例如根据模板频率或某种先验排序取前 64 个模板），保证搜索仍然有可探索的动作空间。

### 6. 库存匹配与 `is_solved` 判定要用统一的 SMILES 规范

- 在 Route 或 evaluator 中，找到判断路线是否 “solved” 的逻辑，例如 **`Route.is_solved(stock)`** 或 **`audit_route`**。
- 问题通常是：
  - 路线终点分子的 SMILES 写法与库存中 SMILES 的规范化方式不一致（芳香式、立体信息、盐型等），导致明明已经拆到库存分子了，却字符串对不上。
- 修复要求：
  1. 定义一个统一的 **`canon_smiles`** 函数，对所有用于匹配的 SMILES 调用：

```
python

def canon_smiles(s):
    mol = Chem.MolFromSmiles(s)
    return Chem.MolToSmiles(mol, canonical=True, isomericSmiles=True) if mol else s
```

  2. 对库存（可购分子）在加载时就做一次 canonical 化，存为集合。
  3. 在 **`is_solved`** 和所有与库存匹配的函数中，统一使用 **`canon_smiles`**。
  4. 可选：对于常见溶剂/盐型（如 **`Cl-`**, **`Na+`**, **`K+`** 等），可以在路线终点做简单的处理（把小离子、溶剂剥离），避免这种细节导致误判。

### 7. GP 搜索与适应度的基本策略（只做轻量调整）

- 不要求你大改搜索算法，只做一些最基本的稳健性修复：
  1. 确保变异与交叉操作不会生成语法非法的 Program。
  2. 确保适应度里 **`solved`** 这个目标/特征在标量化时权重足够大，优先鼓励找到任何一条可行路线（哪怕是多步的）。
  3. 保留多目标（如 route_len, SCScore, purchasable_fraction）作为次要排名依据。

---

## 需要达成的“验收标准”（至少要做到）

1. 在一个简单“玩具世界”上（例如：
    - 目标：一个简单分子，如 **`CCO`**；
    - 模板：一个单步模板能把它拆成两个库存分子；
    - 库存：包含这两个反应物），
      运行 **`run_gp_search_demo.py`** 或相应脚本时：
    - 至少能看到若干个体的 **`solved`** 从 0 变为 1；
    - 能输出至少一条完整、语法合法的路线（Route），其 **`route_len >= 1`** 且终点全部在库存中。
2. 在真实数据脚本上（例如 **`run_real_data_gp.py`** 或你找到的真实管线脚本），
    - 就算 solved 比例暂时仍然不高，日志中也能清晰看出每一次失败的主要原因是：
        - 模板不匹配？
        - 口罩为空？
        - Program 结构非法？
        - 还是化学引擎本身的问题？

---

# 其他约束

- 请尽量保持现有的项目结构和主函数接口不变（例如 **`run_gp_search_demo.py`** 的命令行参数等），在此基础上修复和增强。
- 对第三方引用工程（**`synga-FE64`**, **`ChemProjector-main`**, **`LLM-Syn-Planner-main`**）不要做大改，只在确有必要时参考它们的实现思路。
- 允许添加少量新的工具函数（如 **`canon_smiles`**, **`canonicalize_unique`**），允许添加适量日志/断言。
- 修改完成后，请在代码中用清晰的注释标记你修复的关键点，方便我之后 review 和继续开发。

---

请你按照以上要求，**自动审查整个代码库**，对相关文件做出必要的修改与重构，使得这个 GP 逆合成项目在逻辑上自洽、执行稳定，并且至少能在简单测试用例上找到一条完整的逆合成路径。