

HOMWORK 8*

AGENTS FOR SCIENTIFIC DISCOVERY†

24-880 AI AGENTS FOR ENGINEERS
<https://cmuagents.github.io>

OUT: March 23rd, 2026
DUE: March 30th, 2026
TAs: Peter Pak
CAs: Hardik Choudhary

Instructions

- **Collaboration Policy:** Please read the collaboration policy in the syllabus.
- **Late Submission Policy:** See the late submission policy in the syllabus.
- **Submitting your work:** You will use Gradescope to submit answers to all questions and code.
 - **Written:** You will submit your completed homework as a PDF to Gradescope. Please use the provided template for the submissions which can be written in $\text{\LaTeX}$ (via [Overleaf](#)) or directly on the PDF document. Each answer should be within the box provided. If you do not follow the template, your assignment may not be graded correctly and there will be a **2% penalty** (e.g., if the homework is out of 100 points, 2 points will be deducted from your final score).
 - **Code:** Gradescope’s Autograder will be used to evaluate the coding portions of this assignment. In order to best obtain the points for this assignment, please follow the code upload instructions. If you do not follow the instructions correctly the autograder may not work properly.

Question	Points
AlphaEvolve (Written)	13
OpenEvolve (Code)	70
Agentic Additive Manufacturing (Written)	12
Programming Scripts Upload	0
Collaboration and Generative AI Questions	5
Total:	100

*Adapted from CMU 10-623 Generative AI Homework Template

†Compiled on Monday 23rd March, 2026 at 20:41

Introduction

In this homework assignment we'll look into the some of the agentic applications within scientific discovery.

At a high-level, this homework will cover the following aspects.

1. Investigate AlphaEvolve and implement OpenEvolve.
2. Explore the agentic additive manufacturing alloy evaluation workflow.

Local Environment

For this homework we'll be using `uv` as the project and package manager. If you haven't install `uv`, you can follow the installation instructions here <https://docs.astral.sh/uv/>. You can explore all of the available commands using `uv --help` but in short, here are a couple of useful commands that you'll often use:

<code>source .venv/bin/activate</code>	(Linux and Mac) Activates python environment.
<code>source .venv/Scripts/Activate</code>	(Windows) Activates python environment.
<code>uv sync</code>	Installs / synchronizes projects dependencies.
<code>uv add <package></code>	Installs <code>pip</code> package and adds it to project dependencies.
<code>uv remove <package></code>	Removes <code>pip</code> package dependency from project.
<code>uv run <script.py></code>	Runs python script inside of <code>uv</code> environment.

1. Navigate to folder project directory and synchronize dependencies with:

```
uv sync
```

2. Activate the environment

Mac and Linux:

```
source .venv/bin/activate
```

Windows:

```
source .venv/Scripts/activate
```

3. Test `uv` works and run `main.py` file.

```
uv run main.py
```

1 AlphaEvolve (Written) (13 points)

Novikov et al. [2025] introduces AlphaEvolve, an evolutionary coding agent that combines large language models with an evolutionary search framework to discover improved algorithms across a broad range of scientific and engineering problems. At its core, AlphaEvolve orchestrates an autonomous pipeline in which LLMs iteratively propose code modifications, automated evaluators score the results, and a program database retains the most promising solutions discovered so far. This loop allows AlphaEvolve to go beyond what a single LLM query could produce, enabling discoveries that require sustained iterative refinement.

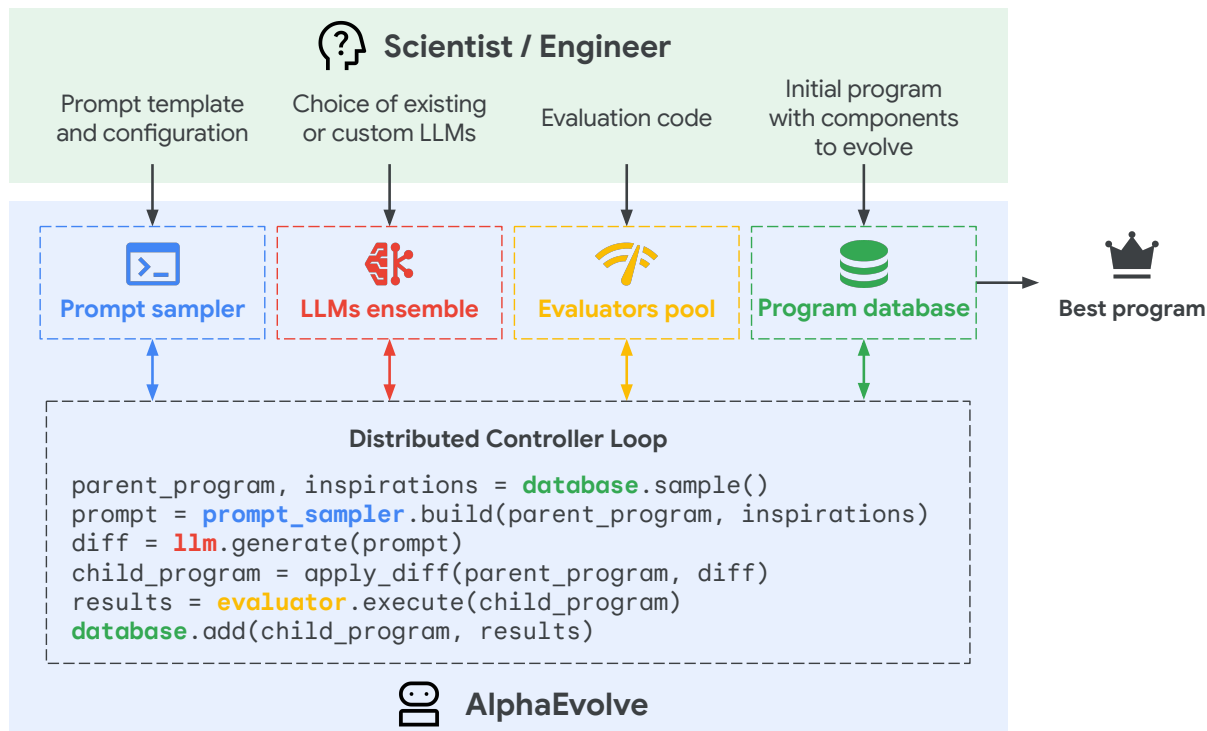


Figure 1: From Novikov et al. [2025], “Expanded view of the AlphaEvolve discovery process. The user provides an initial program (with components to evolve marked), evaluation code, and optional configurations (Section 2.1). AlphaEvolve then initiates an evolutionary loop. The Prompt sampler uses programs from the Program database to construct rich prompts (Section 2.2). Given these prompts, the LLMs generate code modifications (diffs), which are applied to create new programs (Section 2.3). These are then scored by Evaluators (Section 2.4), and promising solutions are registered back into the Program database (Section 2.5), driving the iterative discovery of better and better programs.”

- 1.1. (5 points) Describe the core evolutionary loop in AlphaEvolve [Novikov et al., 2025]. In your answer, identify at least **three main components** of the system and explain how each contributes to the agent's ability to discover improved solutions.

- 1.2. (2 points) According to [Novikov et al. \[2025\]](#), how does the LLM in AlphaEvolve generate new candidate programs during the evolutionary process?
- ☐ It generates entirely new programs from scratch in each iteration.
 - ☐ It generates targeted modifications (e.g., diffs or edits) to existing programs drawn from the population.
 - ☐ It applies fixed, domain-specific mutation operators defined by human researchers.
 - ☐ It fine-tunes its own weights using gradient feedback from the evaluators.
- 1.3. (2 points) Which of the following best describes the role of **evaluators** in AlphaEvolve [[Novikov et al., 2025](#)]?
- ☐ They are human domain experts who judge the quality of each proposed algorithm.
 - ☐ They are automated functions that score programs and provide feedback used to guide the evolutionary search.
 - ☐ They are separate neural networks trained to predict whether a program is correct.
 - ☐ They are the same LLMs used for code generation, self-evaluating their own outputs.

- 1.4. (4 points) Beyond mathematics, [Novikov et al. \[2025\]](#) applies AlphaEvolve to Google's computational infrastructure. List **two distinct application areas** and briefly describe the improvement achieved in each.

2 OpenEvolve (Code) (70 points)

OpenEvolve is an open-source reimplement of the AlphaEvolve framework. In this question you will build the complete evolutionary loop from scratch. The starter code is in `./programming/open_evolve.py`.

Fill out all `# Fill in` sections. A demo is provided in the `if __name__ == "__main__":` block to help you develop and debug.

2.1. (14 points) Implement the `ProgramDatabase` class.

1. `add(program, score)` — append the program string and its score to the internal storage.
2. `sample(n, temperature)` — return `n` `(program, score)` tuples sampled **with replacement** using softmax-weighted probabilities:

$$w_i = \frac{\exp(s_i/\tau)}{\sum_j \exp(s_j/\tau)}$$

where s_i is the score of program i and τ is the temperature. Return an empty list if the database is empty.

3. `best()` — return the `(program, score)` pair with the highest score.
4. `top_k(k)` — return the `k` highest-scoring `(program, score)` pairs sorted descending. If `k` exceeds the database size, return all entries.

2.2. (6 points) Implement `build_prompt(task_description, inspiration_programs)`.

The prompt must include: (1) an introductory instruction, (2) the task description, (3) each inspiration program with its score inside a ````python` code block, and (4) a closing instruction asking the LLM to output an improved program inside a ````python` block.

2.3. (4 points) Implement `parse_program(response)`.

Extract the content between ````python` and ````` markers (case-insensitive) from the LLM response. If no such block is found, return the full response stripped of leading/trailing whitespace.

2.4. (16 points) Implement `evolve(task_description, seed_program, evaluator, llm, ...)`.

Follow the algorithm described in the docstring:

1. Initialize (or reuse) a `ProgramDatabase`.
2. Evaluate the seed program and add it to the database.
3. For each iteration: sample, build prompt, query LLM, parse, evaluate (catching any exception as score `0.0`), and add to database.
4. Return `db.best()`.

2.5. (6 points) Implement `temperature_schedule(iteration, n_iterations, start_temp, end_`

Return the softmax temperature at the given iteration using **linear annealing**:

$$\tau_t = \tau_0 - (\tau_0 - \tau_{\min}) \cdot \frac{t}{N - 1}$$

where $\tau_0 = \text{start_temp}$, $\tau_{\min} = \text{end_temp}$, $t = \text{iteration}$, and $N = \text{n_iterations}$. At $t = 0$ the result is `start_temp`; at $t = N - 1$ it is `end_temp`.

2.6. (24 points) Implement `island_evolve(task_description, seed_program, evaluator, llm,`

Run `evolve()` independently on `n_islands` fresh `ProgramDatabase` instances, each seeded with `seed_program`. After all islands finish, return the `(program, score)` with the highest score across all islands.

You can run and test the code with the following commands:

<code>uv run programming/open_evolve.py</code>	Runs the demo (useful for development)
<code>uv run test.py open_evolve</code>	Runs the test suite used by the autograder

The provided tests are the exact ones that will run on the autograder (i.e. if your local tests pass, they should also pass on the autograder).

3 Agentic Additive Manufacturing (Written) (12 points)

Pak et al. [2026] presents a multi-agent system that automates alloy evaluation for additive manufacturing (AM) using Large Language Models. The system is built around three specialized Model Context Protocol (MCP) servers orchestrated by Claude Sonnet: a *workspace-agent* that manages file-based state across tools, a *thermo-calc* server that computes thermophysical properties via CALPHAD methods, and an *additive-manufacturing* server that generates lack-of-fusion (LoF) process maps using the Rosenthal heat-source equation. Together they allow the agent to autonomously evaluate both established and novel alloy compositions without continuous human intervention.

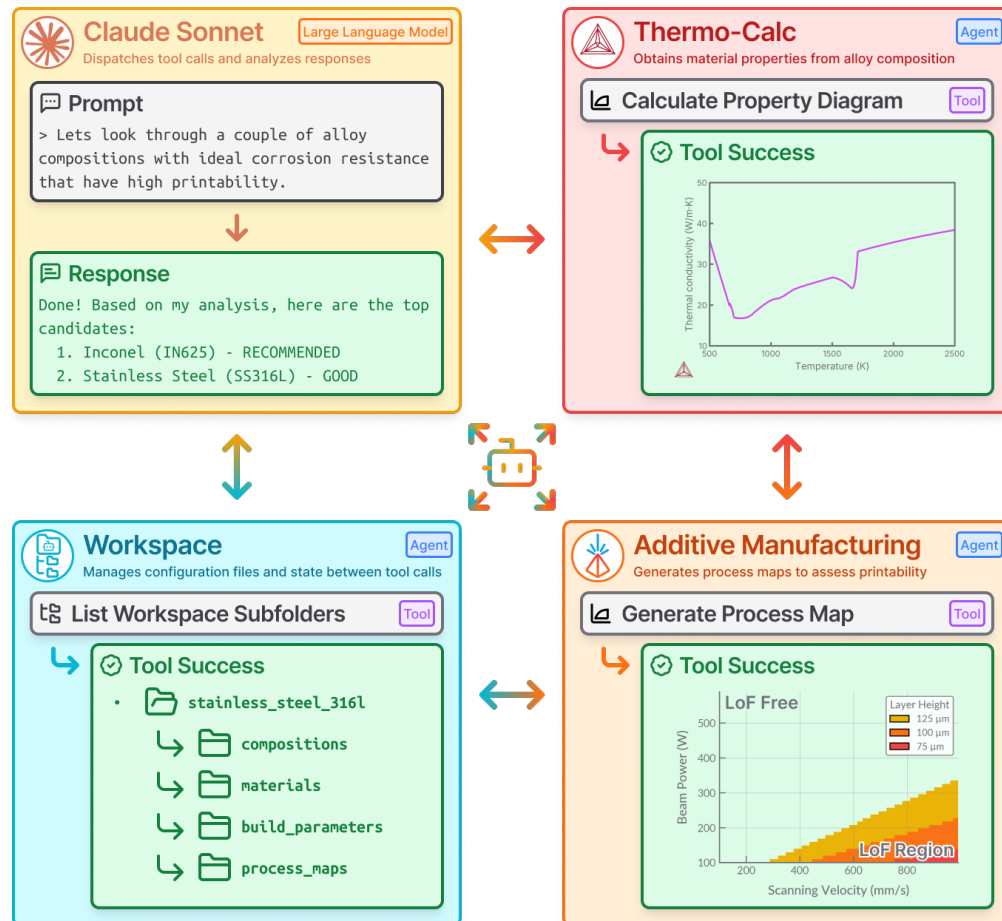


Figure 2: From Pak et al. [2026] “(Top Left) An input query for alloy compositions regarding the printability of an additively manufactured part suitable for its intended use case is provided to Claude Sonnet. This Large Language Model (LLM) calls the tools necessary to generate and analyze each potential alloy compositions, providing a response of candidates ranked by their content of their lack of fusion fusion regimes. (Top Right) Thermo-Calc allows for the retrieval of material properties for an hypothetical alloy composition, for instance thermal conductivity, to be used in down stream lack of fusion calculations. (Bottom Left) Workspaces provide a way for each of the tools to effectively communicate with one another and handles state management and file organization. (Bottom Right) Tools managed by the Additive Manufacturing subagents then utilize the calculated material properties from the Thermo-Calc subagent to generate a lack of fusion process map to send back to the LLM for analysis and final recommendation.”

- 3.1. (4 points) What is the Model Context Protocol (MCP), and why does [Pak et al. \[2026\]](#) use it rather than having the LLM call Python functions directly? In your answer address the key design principle the authors describe as “separation of concerns.”

- 3.2. (3 points) Pak et al. [2026] evaluates alloy printability by generating a **lack-of-fusion (LoF) process map**. What does the LoF process map show, and what does the LoF criterion predict about a given set of process parameters?

- 3.3. (5 points) Identify **two limitations** of the Rosenthal equation as used in [Pak et al. \[2026\]](#), and for each explain how it might affect the accuracy of the generated process maps.

4 Programming Scripts Upload (0 points)

To upload your code to Gradescope, run the following command to generate a zip file:

```
uv run programming/zip_programming.py
```

Upload the generated zip file to the appropriate programming slot on Gradescope.

4.1. Make sure the following files are included when uploading to Gradescope

- `open_evolve.py`

5 Collaboration and Generative AI Questions (5 points)

After you have completed all other components of this assignment, report your answers to these questions regarding the collaboration policy. Details of the policy can be found in the syllabus.

- 5.1. (2 points) Did you collaborate with anyone on this assignment? If so, list their name or Andrew ID and which problems you worked together on.

- 5.2. (3 points) Did you use generative AI for parts of this assignment? If so, include the input prompts along with links to the conversation.

References

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. AlphaEvolve: A coding agent for scientific and algorithmic discovery, June 2025. URL <http://arxiv.org/abs/2506.13131>. arXiv:2506.13131 [cs].

Peter Pak, Achuth Chandrasekhar, and Amir Barati Farimani. Agentic additive manufacturing alloy evaluation. *Additive Manufacturing Letters*, 17:100355, April 2026. ISSN 2772-3690. doi: 10.1016/j.addlet.2026.100355. URL <https://www.sciencedirect.com/science/article/pii/S2772369026000022>.