UNCORRELATED ORBITAL TRACK PROCESSING

EARLY STATUS REPORT

Kyle F. Galang, Aurela Broqi, Ruben Dennis, Aaron Nogues, Ezra Stone

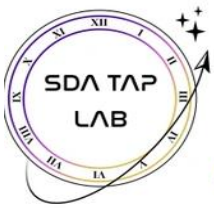
COP4934: Computer Science Senior Design I

Dr. Richard Leinecker

University of Central Florida

Orlando, Florida

October 03, 2025



AFRL

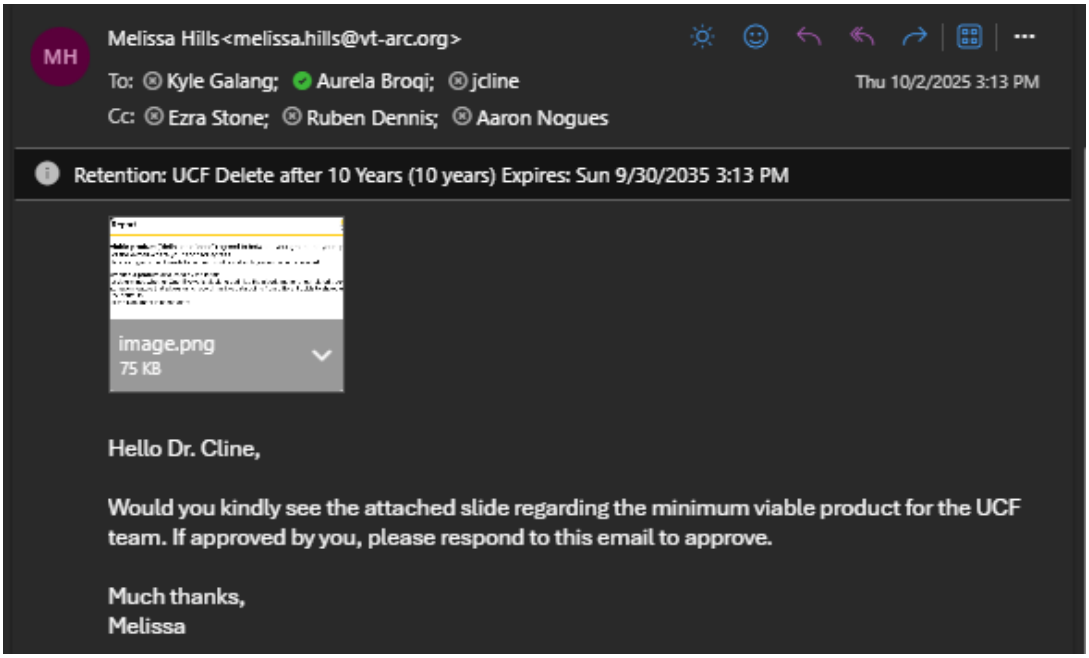
Minimum Viable Product

For our minimum viable product, we discussed with the sponsors in a meeting what they wanted out of our timeline of senior design 1 and 2. Connecting things together from previous meetings, our team discussed what they might have wanted before the meeting, and we created 4 minimum viable products that all connect with one another. The first one being a full research document that determines whether OpenEvolve is viable to optimize the algorithms they are facing for uncorrelated track processing. Second being a poster summarizing our findings in OpenEvolve and showcasing our data points in charts and graphs from their specified colored schemes. These first 2 minimum viable products are just what the AirForce Research Laboratory wants. But since we are in senior design, we need to fulfill the requirements of what makes a project senior design worthy. Which includes software design and development. So, our team brought up that requirement to make this project senior design worthy to our sponsors. Where we added two more minimum viable products, the third one being some type of automation script that we have to develop in order to get all the data points we run through when testing such algorithms. Lastly, the fourth one being a graphical user interface makes the process and methodology of using OpenEvolve simpler for our target audience who aren't so familiar with the program.

Early Status Report

College of Engineering and Computer Science
UNIVERSITY OF CENTRAL FLORIDA

- A **minimum viable product** ("definition of done") agreed to between your group and your sponsor. Include a screenshot of the e-mail where your sponsor agrees.
  - You can include stretch goals, but it needs to be clear what is a stretch goal and what is required.
- Current **minimum viable product** discussed by the team:
  1. Document that determines whether OpenEvolve is viable to optimize the algorithms for uncorrelated track processing.
  2. Create an automation feature that allows us to record multiple data points from different LLMs to showcase the different categories and their outputs.
  3. Poster containing a data table and summary of the research on OpenEvolve.
  4. Graphical User Interface Development.



During the meeting, the AirForce Research Laboratory wanted deliverables of documentation and a poster for the research. But we also discussed senior design requirements for software development and design to make it worthy of senior design. They agreed with us and also helped fix some of our minimum viable product points during the meeting. We asked for the email confirmation, but our sponsor forwarded it to Dr. Cline who helped us create the points, but we still have not gotten a reply.

Our team is planning on creating a graphical user interface and automation scripts for data collection in order to meet senior design requirements since we can’t just do research doing senior design 1 and 2.

Description of what you have accomplished so far

General Team Description:

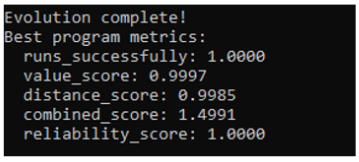
Currently for the past weeks we have been trying to organize how the project goes and what our deliverables are from our sponsors and how they correlate to senior design. We've been able to set up OpenEvolve on everyone's computer, where we were able to run the example programs just like the sponsor asked. But afterwards we were lost since the sponsors did not really tell us exactly what they wanted from OpenEvolve. The weekly meetings are very helpful since they give a lot of clarification of what the sponsor is looking for. We found out that their main delivery of what they want is strictly research documentation and posters. But we told them about senior design requirements of the team needing to design and develop software to make it senior design worthy. So overall it is a lot of organization and planning on what to do in regard to senior design requirements and what they want from our team. We've set up our documentation for our 20 pages per person, set up our weekly power points to present to our sponsor in our meetings, set up our GitHub, set up our JIRA backlog and sprints, set up 2 weekly meetings for our team, and overall just structuring the project to be successful since there is a lot of flexibility and vagueness to the end goal. We've also submitted our passports to our sponsor so that they can clear everyone as a US citizen.

Kyle Galang's Description:

Most of my work is getting a structure going for our team. We just found out that our sponsors want a more research heavy deliverable consisting of just turning in documentation and posters for the project. But we know that does not cut it for senior design efforts. So, I've been structuring our team to be able to do the work and also know what they need to do as a project manager. Getting everyone caught up with the OpenEvolve set up, setting up deadlines to keep everyone on track, and organizing things to make it easier. I've set up the meetings, GitHub, documentation, powerpoints, and also just dealing with getting as much information from the sponsor. While also collaborating with the team of what their goals are for this project. Of course, we are doing research, but we also need to come up with something as a team that everyone is willing to work on. We currently are in agreement with a full stack development for a graphical user interface to make using OpenEvolve a lot easier. Since our target audience is a population that does not know too much about OpenEvolve. We are also thinking about creating a script that allows us to record multiple data points when running through the iterations of the problems. We've talked about these requirements to the sponsor, and they have agreed onto them alongside their requirements. I believe some future work that needs to be done fairly soon is really understanding what we want to show in the user interface and seeing what tech stack to use to be able to pull from the data and automatically create. One thing we are also working on right now is seeing how the evaluation script is being created based on the algorithm we are trying to optimize. I've ran OpenEvolve multiple times to see how the initial function is being manipulated and at the moment I am working on finding the threshold of the optimizer to see where we can cut off the iterations to save computing power and time.

Time runs:
50 Iterations: approx. 5 min 30 sec.
100 iterations: approx. 12 min.
1000 iterations: approx. 1 hr+.

Examples run:
Function minimization
Circle packing
Signal Processing



Ruben Dennis' Description:

I've gotten the Open Evolve environment running on my personal system. The initial setup wasn't working right away though, Open Evolve repeatedly failed to properly call the Gemini API and entered a loop where it kept trying to ping the service. After some troubleshooting, we found out that the issue came down to how the API key was being referenced. On Gemini's website, the key is provided under its own variable name, but Open Evolve expects the variable to be named OPENAI_API_KEY. When I changed it to the proper name it ran smoothly.

We also ran into another issue with Gemini's student tier, the API doesn't properly use the tokens. This limits how well Open Evolve can interact with it. Since I personally have a paid subscription for ChatGPT, I've been exploring whether there is a way to leverage that access to ping the API for the paid version instead. This may become important if we want consistent and some higher quality responses for larger test cases.

On the experimentation side, I have been able to run the provided example problems, including Function Minimization and AlgoTune Optimization, across several different AI models: Gemini 2.5 Pro, Gemini 2.5 Flash Lite, and GPT-5. At this early stage, I haven't noticed significant performance differences between the models, though that may change once we feed them larger or more complex datasets. For now the outputs look fine and stable, but the lack of variation leads me to wanting to make some more complex problems.

Another thing we brought up to the sponsor is to create a script that automatically documents the different iterations of the "evolved" algorithms and their evaluation results. Open Evolve already produces these iterations as part of its workflow, but right now they are not logged in a structured way. The goal is to checkpoint each version of the algorithm, along with its evaluation metrics, and store them in a CSV file for analysis.

Ezra Stone's Description:

In the past few weeks I have been familiarizing myself with the AlphaEvolve paper and the OpenEvolve framework. I've read through the 44 page AlphaEvolve paper to help me have a stronger understanding of the theoretical foundation behind the system. From this I have been able to understand the process used to evolve these programs, and the kinds of problems it is designed to solve. This gave me a clearer picture not just of the technical details, but also how AlphaEvolve/OpenEvolve approaches optimization problems in an adaptive way.

After reading through the paper I successfully set up OpenEvolve on my system and verified the installation by running one of the provided example problems. That being the circle packing problem, in this problem things such as boundaries, packing density, and preventing overlaps are important for the code's evolution. Running this example showed me how OpenEvolve structures a problem, how the solution evolves over multiple iterations, and how the results can be interpreted. This example was important because it confirmed that my system is properly configured and that I can now focus on exploring future problems rather than troubleshooting setup issues.

Moving forward, I plan to continue working through the provided example problems available within OpenEvolve. Each example highlights different aspects of the framework, and by working through them I should be able to gain a deeper understanding of the system's potential applications. This has also allowed me to draw clearer connections to our project and begin considering how the same methods can be adapted to address the challenges we may face when the building process begins.

In addition to the technical exploration, our team has also been actively working on sorting out roles and responsibilities for the project. We have spent some time discussing how to divide the work so that each member can focus on their strengths, while also ensuring that no single person is overloaded. So far we have been able to decide who will be project manager and who will be our scrum master. In this coming week we should be assigning roles for the rest of the group, as well as who can work to fill in roles when a member may be missing. Having

these roles clearly defined will make our workflow more organized, reduce redundancy, and allow us to work more efficiently as a team.

Finally, we discussed with our sponsor the idea of building a script to automatically document the different iterations of the evolved algorithms. This being things such as the time taken for the evolution to happen along with other corresponding evaluation results. At this time the outputs are not stored in a structured or easily analyzable format through OpenEvolve. In realizing this we came up with the idea to checkpoint each version of the algorithm and capture its evaluation metrics in a CSV file. By doing this we can generate a clear record of progress over time and understand when and what algorithms work best through it. This logging would also take into account the niceness level in terms of our prompts . Ultimately, it would provide both transparency and a data driven way to evaluate the effectiveness of different practices within OpenEvolve.

Aaron Nogues' Description:

After viewing and running the examples from the github, it was determined that the evaluator script, while having similar outputs into open evolve, has logic that is intimate with the problem itself. Thus it is determined that any algorithm developed to be evolved has to be relative to the issue itself, which must be hand coded. It would be much more convenient if there was some sort of automatic evaluator writing script, especially since there are already many variables introduced into the system to modify performance of the output. This would extend the categories of data we are looking for. From these:

- Base Data(data of our resources prior)
- Context Amount (how much context fed to the AI affects the efficiency of the output)
- Context Quality (how much of the context fed to the AI is actually related to the problem)
- Human Format of Prompting (how humans may or may not prompt the AI)
- Base LLM to LLM Comparison (given the same variables across the board how do other LLMs hold up against each other)
- Human and Machine Language Compatibility (How does changing the human language, vocabulary and programming language affect quality)
- Problem Type (how does open evolve handle types of problems)
- Evolution vs Knowns (How does the quality of the output of open evolve hold up against the current known quality).

To also include variables for the evaluator script, which have yet to be determined. The current intent is to formulate the methods of creating an evaluator script and then manually implement them. Automation of this evaluator script creation after speaking with the sponsor was determined to be a stretch goal.

For the current categories the measurement for each of these should be simple, as the evaluator script must already compare correctness and speed across each iteration. However for the categories that relate to the prompt, a custom script will have to be created to handle testing different prompts across the same instance for convenience.

Aurela Broqi's Description:

As of right now, I began working with the AlgoTune benchmark suite inside OpenEvolve. Cloned the repository and installed all required Python dependencies inside the virtual environment. Created and activated the virtual environment to keep the dependencies contained. Did unit testing and baseline runs to verify the installation. I successfully ran initial tasks such as function minimization and produced results.

My current effort is focused on preparing and running AlgoTune tasks such as polynomial real, convolve2d˙fullfill, and working on testing attention optimization to evaluate performance improvements.

Right now, my focus is on setting up and running OpenEvolve experiments within the AlgoTune benchmark suite. I cloned the repository, created and activated a virtual environment, and installed all required dependencies. I verified the installation by running unit tests and baseline examples. I still need to test other examples (e.g., robust regression in R, symbolic regression on LLM-SRBench) and document both the environment setup steps and how the model behaves during evolution.

My work so far has been directed toward understanding how OpenEvolve evolves algorithmic solutions across tasks, starting with smaller examples like function minimization and moving toward more complex AlgoTune optimizations.

Beyond benchmark tasks, I also discovered that OpenEvolve can solve competitive programming problems such as those hosted on Kattis. For instance, in the Alphabet problem, the system started from a incorrect solution and, after a few iterations, evolved into a fully correct dynamic programming approach. This demonstrates OpenEvolve's ability to discover well-known algorithms such as the Longest Increasing Subsequence (LIS), and validates its generality beyond just optimization benchmarks. Evolved Algorithm Example (Kattis: Alphabet)

```python
import sys

for line in sys.stdin:
    s = line.strip()

n = len(s)
dp = [1] * n

for i in range(1, n):
    for j in range(i):
        if s[i] > s[j]:
            dp[i] = max(dp[i], dp[j] + 1)

longest_alphabetical_subsequence_length = max(dp)
ans = 26 - longest_alphabetical_subsequence_length
print(ans)
```

This finding highlights that OpenEvolve can use online judge platforms as a “fitness function” to evolve correct and efficient algorithms from scratch. The workflow uses Kattis’s submission pipeline, where a helper script (submit.py) automates the process of sending source code to the Kattis judge. This script reads the command line arguments (solution file names), infers the problem ID and language from extensions (e.g., .py → Python 3, .cpp → C++, .f90 → Fortran), logs in using the local .kattisrc credentials, and posts the submission via HTTP. The server then compiles and runs the solution against hidden test cases , while the script continuously polls the results and displays progress back in the terminal. In effect, Kattis itself acts as the evaluation harness while OpenEvolve drives the code mutations until a passing solution emerges.

In addition to online judge testing, OpenEvolve is able to improve symbolic regression models by evolving actual Python code that contains explicit mathematical formulas. This means that you can open the initial program.py file at any generation and directly observe how the code changes as the algorithm improves. For example:

Initial generation:

```python
def func(x, params):
    return params[0]*x[:,0] + params[1]*x[:,1]
```

After a few generations:

```python
def func(x, params):
    return -params[0]*x[:,0] - params[1]*x[:,0]**3 + params[2]*np.sin(x[:,1])
```

Later generation:

```python
def func(x, params):
    return -params[0]*x[:,0] - params[1]*x[:,0]**3 - params[2]*x[:,0]*x[:,1] +
    ↪  params[3]*np.sin(x[:,1])
```

This makes the evolutionary process transparent, since you can literally watch a naive linear model transform into a nonlinear expression with polynomial, interaction, and trigonometric terms that better capture the dataset.

This complements my current work with AlgoTune benchmarks and strengthens the foundation for using evolutionary approaches in both performance optimization and general algorithm discovery.

Evolution Summary

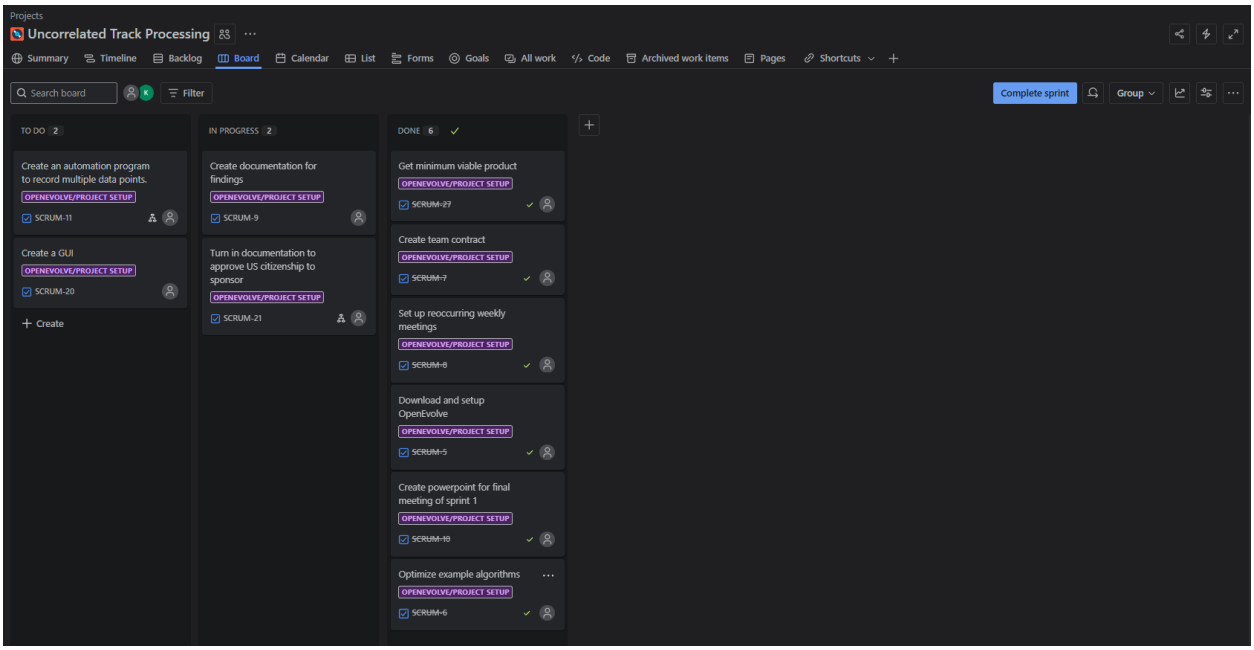
```
Evolution complete!
Best program metrics:
  runs_successfully: 1.0000
  value_score:       0.9418
  distance_score:    0.7566
  combined_score:    1.2148
```

Checkpoint Information

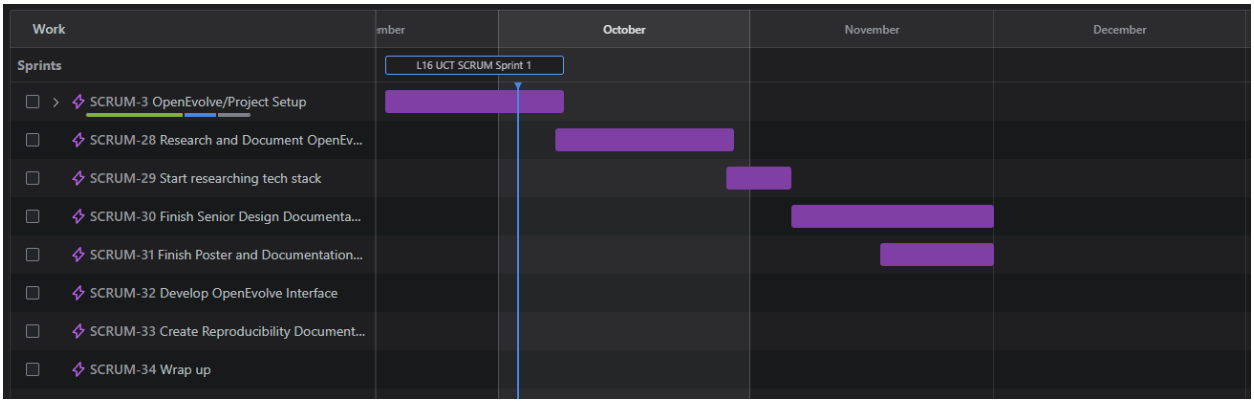
```
Latest checkpoint saved at:
examples/function_minimization/openevolve_output/checkpoints/checkpoint_100

To resume, use:
--checkpoint examples/function_minimization/openevolve_output/checkpoints/checkpoint_100
```

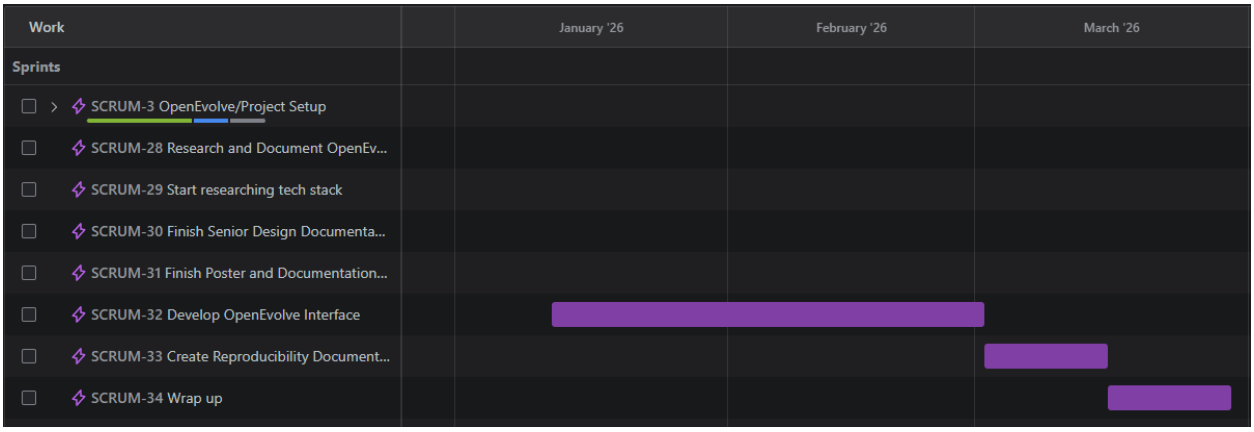
Notional Sprint Map

Current Sprint:



Notional sprint map from now to the end of March:





Sprints short descriptions:

Sprint 1 (9/17-10/08) - OpenEvolve/Project Setup: Setting up the project requirements and tying them along with senior design requirements. Organization efforts for scheduling and tasking for the team. On boarding with the Air Force Research Laboratory.

Sprint 2 (10/08-10/29) - Research and Document OpenEvolve: Start researching OpenEvolve based off of what AFRL requirements and start official documentation. Which will tie into our 20 pages per team member requirement in senior design.

Sprint 3 (10/29 – 11/05) - Start Research Tech Stack: Since the requirement of senior design is to be able to design and develop software, we decided to implement an interface to make using OpenEvolve easier. So this sprint is solely focusing on finding the tech stack and researching and documenting our process for senior design requirements.

Sprint 4 (11/06 – 11/30) - Finish Senior Design Documentation: This sprint is solely trying to finish the 20 pages per group member requirement for the end of senior design 1.

Sprint 5 (11/17 – 11/30) - Finish Poster and Documentation: Since we are finished the tech stack research and OpenEvolve research, we need to wrap up the poster and documentation AFRL is asking for so we can focus on strictly developing for senior design 2.

Start of Senior Design 2:

Sprint 6 (01/12 – 03/01) - Develop OpenEvolve Interface: This is the start of senior design 2, since we have all the preliminary research and documentation, we created in senior design 1. We are solely focusing on developing the software in this 7-week sprint.

Sprint 7 (03/02 – 03/15) - Create Reproducibility Document for Software Development: This 2-week effort is to write documentation for the future team that will continue our work and how to use the interface.

Sprint 8 (03/16 – 03/29) - Wrap up: This will focus on presentation practice.