Problem Statement and Goals
SE 4G06

Team 3, Tangle
Calvyn Siong
Cyruss Allen Amante
Edward Gao
Richard Li
Mark Angelo Cruz

Table 1: Revision History

Date	Developer(s)	Change
9/21/2024	Calvyn Siong	Added changes to stakeholder, environment and goals section
9/22/2024	Richard Li	Added Problem Statement section
9/23/2024	Richard Li	Made suggestions to Calvyn's sections
9/23/2024	Calvyn Siong	Implemented feedback
9/23/2024	Richard Li, Calvyn Siong	Added reflection section
...	...	...

1 Problem Statement

1.1 Problem

Tangled Program Graphs (TPG) is a reinforcement learning framework being developed by McMaster's Creative Algorithm Lab under Dr. Stephen Kelly. The main scope of the research is to eventually apply genetic programming principles to embedded systems and to share this framework as an open source library. Reinforcement Learning algorithms have the same underlying purpose: determining the right action given the current state of the environment. The cost to train deep learning neural networks is a huge blocker to democratize access to this technology. TPG aims to solve the cost issue through evolution. However, TPG has two notable problems: software engineering practices, and integrations into more advanced real-world environments.

Despite its innovative approach, TPG's code base does not fully adhere to standard software engineering practices commonly found in large open-source libraries. For instance, it lacks comprehensive documentation, robust testing, and CI/CD, which can pose challenges for researchers or community contributors attempting to onboard or make changes to the framework. With more focus on software engineering best principles (SOLID, abstractions, etc.), the code base can evolve over time, allowing researchers to focus more on actual results, rather than dealing with technical debt.

Currently, TPG has been integrated and validated within fully observable mini-game environments (such as Atari, Pac-man, CartPole etc.). However, real-world scenarios are typically partially observable and dynamic, where an AI agent must adapt to constantly changing conditions. To advance Dr. Kelly's research, testing and validation in more complex, dynamic environments is essential. The ultimate objective is to develop an evolutionary reinforcement learning framework suitable for embedded systems. Achieving this goal requires moving beyond controlled environments and applying evolutionary reinforcement learning to real-world scenarios, further expanding the scope and pushing the boundaries of this research.

1.2 Inputs and Outputs

The problem is sub divided into two main problems: **Adhering to Software Engineering Practices** and **MuJoCo** integration.

1.2.1 Adhering to Software Engineering Practices

When developing code according to best software engineering practices, the **codebase** is the main input. However, there are many different scenarios that need to be encompassed: DevOps integration, code refactoring, and extensive documentation. The main difference is when these scenarios are triggered. For instance, DevOps pipelines are typically run after code has been committed/merged (depends on the requirements specified). Each of these scenarios involves performing transformations on the code base, allowing the team to allow the TPG code base to adhere to the standard open source software development standards. Other sources of inputs could be references that the team can model these software development practices after such as large scale open source projects (i.e., MuJoCo library from DeepMind) to see what standards they adhere to.

When development is complete, the TPG codebase will have a **CI/CD pipeline** so whenever changes are made to the codebase - automatic testing, code checks, and possible deployments are performed allowing for more robust, and higher quality code that everyone can contribute to in a standardized manner. Unit testing allows current code to be validated, ensuring the blocks of code can adhere to many different cases both obvious and edge cases. The output of more refined testing is robust code that has been validated. Documentation

will allow new contributors to understand how to contribute and also onboard to the project in an easy and clear manner.

1.2.2 MuJoCo Integration

For this integration to be successful, TPG and MuJoCo libraries need to be connected together. MuJoCo is an open source physics engine developed by Google DeepMind and enables accurate real-life simulation experiments to be performed for machine learning scientists. The development of the interface requires all of these components as when an experiment is simulated, the interface needs to know what it needs to be simulating on the MuJoCo engine.

A successful interface enables the TPG framework to be evaluated in more dynamic environments on MuJoCo. As a research team, Dr. Kelly is able to perform more "real-world" like experiments and to evaluate the performance of TPG and fine tune the framework to iteratively improve its capabilities.

1.3 Why is this Problem Important?

Current deep learning neural networks, particularly those used for reinforcement learning, are highly inefficient and resource-intensive, making them impractical for many real-world applications, especially in embedded systems and robotics. Addressing this inefficiency is crucial to making intelligent robotics more accessible and cost-effective. The alternative utilizing concepts can accelerate the transition to adopting this technology in embedded systems. This approach could drastically reduce the GPU training costs associated with deep learning, enabling more scalable and adaptable solutions for embedded systems and real-time robotics applications. This transition is critical for making advanced robotics more accessible and economically viable.

Achieving this next stage requies significant effort and adjustments. Firstly, the current codebase needs to adhere to traditional software engineering standards to allow for the next steps of evolution. If more researchers want to contribute or build on top of the framework to tune it to their specific applications, then having robust CI/CD pipelines, unit testing, and extensive documentation are critical for the developer experience. Making the codebase more accessible to all while ensuring the business logic is sound is critical to the future of this problem.

Many existing reinforcement learning models have been tested primarily in "toy" environments—simple, fully observable scenarios like mini-games. While these simulations provide a controlled testing ground, they fail to represent the complexity of real-world environments, which are dynamic and only partially observable. To advance Dr. Kelly's framework and make it applicable to real-world robotics, further experimentation is required in more realistic settings where agents must learn from incomplete information and adapt to changing conditions over time.

Dr. Kelly's research addresses a critical bottleneck in how robots learn to interact with their environment. By providing a more efficient and cost-effective

alternative to current deep learning models, this work has the potential to revolutionize the industry. However, before this solution can be fully integrated into real-world robotics, rigorous testing and validation in dynamic, real-world environments must be conducted. The work ahead is challenging, but the potential impact is transformative—ushering in a new era of intelligent, accessible, and adaptable robotics.

1.4 Stakeholders

1. **Primary Stakeholders: Dr. Stephen Kelly's TPG Research Team**

   Dr. Kelly and his research team are the primary stakeholders of this project. As the original developers of the TPG framework, they are directly invested in and benefit from improving the framework's software engineering practices and expanding its applicability to more complex environments. The success of this project will directly contribute to their ongoing research efforts in reinforcement learning (RL) and genetic programming. The additional integrations created for TPG will allow the group to more efficiently explore new research avenues, including the application of TPG in robotics and advanced simulations.

2. **Secondary Stakeholders: The GEGELATI Team**

   The GEGELATI team, a French-based TPG research organization, is a secondary stakeholder. They are adopting the TPG algorithm for research purposes that differ from Dr. Kelly's project. Dr. Kelly has stated that this group references his TPG project for their own research purposes. While they are not directly involved in the development of this project, they stand to benefit from its outcomes. Improvements in the TPG framework, particularly in terms of software reliability and scalability, will provide valuable insights and tools that can be incorporated into their research. The establishment of an interface between TPG and modern robotics simulators may also open up new possibilities for their applications of the TPG algorithm.

3. **Tertiary Stakeholders: The Wider Reinforcement Learning (RL) Community**

   The broader reinforcement learning community is a tertiary stakeholder in this project. Researchers and developers interested in the intersection of evolutionary algorithms and machine learning may benefit from the enhanced TPG framework. The improvements made through this project, including standard software engineering practices and expanded testing environments, will increase TPG's accessibility and usability. Additionally, the publication of project results in international conferences could raise awareness of TPG's capabilities, potentially encouraging wider adoption and collaboration within the RL community.

1.5 Environment

The capstone project will primarily be based on the existing TPG framework environment, which will consist of the existing C++ codebase which will be refactored and modified to comply with standard software engineering principles, such as continuous integration, automated testing, and improved documentation. In addition, a key objective of the capstone is to extend the TPG framework beyond its current application in Atari video games by integrating it into **MuJoCo (Multi-Joint dynamics with Contact)**, a widely used physics engine developed by **Google DeepMind**. MuJoCo provides highly accurate physical simulations, particularly suited for robotics. By interfacing TPG with MuJoCo, the project aims to evaluate the performance of the framework in controlling complex robotic systems, such as quadrupeds. This integration will allow for more realistic testing environments, expanding the boundaries of TPG's current capabilities.

2 Goals

1. **TPG Framework must adhere to a modern software engineering Dev Ops pipeline**

   Set up fully automated CI pipeline to ensure all code changes in the TPG framework are continuously tested, verified and follow proper guidelines

   Measurable Outcomes: To accomplish this goal, there should be a complete integration of a CI tool such as Github Actions or Gitlab CI. The CI pipeline should be invoked when changes are made to the framework via new PRs and should not result in blocking errors (either accept or reject changes after CI pipeline analyzes changes).

   Reasoning: Establishing a CI pipeline will standardize software development workflows, reduce human error, and improve code reliability. It ensures that the framework is stable with each update, promoting scalability and maintainability

2. **The TPG Framework should be well tested**

   Develop a comprehensive testing suite for the TPG framework, including unit tests, to ensure major components are thoroughly validated and code coverage is maximized.

   Measurable Outcomes: To accomplish this goal, unit and integration tests will be implemented for core TPG functions, ideally hitting at least 80% code coverage (subject to change after further analysis). The tests must be able to be run as part of the CI pipeline, ensuring that all functionality remains valid after code changes.

   Reasoning: Introducing a well-structured testing suite is critical for ensuring the reliability and correctness of the TPG framework. Unit tests will

verify that individual components behave as expected, and adding integration tests will confirm the system functions correctly as a whole. High test coverage ensures that potential issues are caught early, reducing future bugs and improving maintainability. This goal also aligns with the project's emphasis on adopting best practices in software engineering.

3. **The TPG framework must have an Agent-Environment Interface between itself and MuJoCo**

   Develop a robust interface that allows the TPG framework to communicate and control robotic agents within the MuJoCo physics engine.

   Measurable Outcomes: To accomplish this goal, the TPG framework should contain a fully functional interface for a basic agent (e.g., a program that controls a simple joint robotic arm) in MuJoCo. By the end of the project, this can be further expanded to having the interface to support more complex agents.

   Reasoning: This interface will enable the TPG framework to be tested in more realistic and complex environments. Successful integration with MuJoCo will validate TPG's performance in a cutting-edge robotics simulator, essential for showcasing its real-world capabilities. Furthermore, the current testing environment for TPG is stationary (all states are known/given), and integrating with Mujoco enables for testing with dynamic environments, where the agents will have to change over time.

4. **The TPG Integration in MuJoCo should have determinable behavior when ran**

   Perform a series of experiments to evaluate the behavior and performance of agents controlled by the TPG algorithm within the MuJoCo environment.

   Measurable outcomes: Analyze and record key metrics such as agent performance, model selection process, task success rate, and behavior patterns in various scenarios.

   Reasoning: Extending upon the previous goal, these experiments will provide insights into how well TPG can control complex agents in a physics-based simulator like MuJoCo. Understanding the agent's behavior in a dynamic environment will be essential for evaluating the practical applicability of TPG, similar to the previous goal.

3 Stretch Goals

1. **The TPG framework is integrated with additional Robotics Simulators**

   Extend the agent-environment interface to support additional simulators

   Reasoning: Supporting multiple simulators will increase the flexibility of TPG, allowing it to be tested in diverse environments. This would expand

the framework's use cases, enhancing its appeal to the wider RL and robotics communities.

2. **The TPG framework is linked to a User-Friendly GUI for Experiment Management**

   Create a graphical user interface (GUI) for researchers to manage, run, and visualize experiments within the TPG framework.

   Reasoning: A user-friendly GUI would make the TPG framework more accessible and easier to use for the stakeholders involved. This could result in more efficient usage of the framework.

4 Challenge Level and Extras

The challenge level of this project is deemed to be **general**, as it builds upon an existing foundation of work developed by Dr. Kelly's research team.

While the project involves expanding the Tangled Program Graphs (TPG) framework and integrating it with new environments, the foundational research, including the algorithmic structure of TPG and its application in simpler environments, has already been established. This provides a stable starting point, reducing the complexity typically associated with new algorithm integrations.

Although the integration of TPG with advanced robotics simulators such as MuJoCo introduces new technical challenges with an unexplored and wider scope (in regards to extending the application to robotics simulations), these are manageable due to the existing knowledge base and the prior successful implementation of TPG in simpler environments like Atari video games. The task primarily involves adapting and extending the current work rather than developing entirely new algorithms.

Additionally, the software engineering improvements, such as implementing continuous integration, introducing a testing suite, and generating comprehensive documentation, follow standard best practices and are well-supported by available tools. These tasks, while requiring careful planning and execution, are considered general development work and are feasible within the scope of the project.

One of the extras to accompany this project will involve implementing **user documentation** to support the newly integrated features and enhancements. This documentation will be designed to guide users through the updated functionalities, including the integration of the TPG framework with the MuJoCo simulator, as well as any improvements made to the codebase and testing suite.

Appendix — Reflection

The purpose of reflection questions is to give you a chance to assess your own learning and that of your group as a whole, and to find ways to improve in the future. Reflection is an important part of the learning process. Reflection is also an essential component of a successful software development process.

Reflections are most interesting and useful when they're honest, even if the stories they tell are imperfect. You will be marked based on your depth of thought and analysis, and not based on the content of the reflections themselves. Thus, for full marks we encourage you to answer openly and honestly and to avoid simply writing "what you think the evaluator wants to hear."

Please answer the following questions. Some questions can be answered on the team level, but where appropriate, each team member should write their own response:

1. **What went well while writing this deliverable?**

   When writing this deliverable, there were several things that went well.

   One of the most notable aspects was the effective communication and teamwork within our group. We made an emphasis to split up tasks in a balanced way, which helped keep the workload manageable and clear. This clear division of labor allowed us to set realistic expectations, including communicating our individual estimated timelines (ETAs) for task completion in our group chats, and ensuring that everyone stayed on track.

   Another key element was our commitment to reviewing each other's work. By frequently reviewing pull requests and providing constructive feedback, we were able to ensure the quality of our submission and catch any potential issues early. This also fostered a culture of openness to feedback, where we valued and incorporated each other's suggestions without hesitation, improving the consistency between our differing writing styles.

   Dr. Kelly's involvement also played a pivotal role in our progress. From the beginning, he took time to explain reinforcement learning concepts to us in a way that deepened our understanding and directly contributed to our ability to contextualize Tangled Program Graphs (TPG) within his larger research goals. His willingness to meet regularly and support us synchronously and asynchronously—gave us the guidance we needed to overcome challenges and stay aligned with the direction of his research.

2. **What pain points did you experience during this deliverable, and how did you resolve them?**

   One of the primary challenges we faced while working on this deliverable was defining clear and precise goals without having sufficient context. Initially, we found ourselves navigating in ambiguity, which made it difficult to establish a solid foundation for our project. This lack of context created uncertainty and hindered our ability to align our objectives effectively with the overarching research aims.

To tackle this issue, Calvyn took the initiative to draft a preliminary set of goals prior to our scheduled meeting with Dr. Kelly. This rough draft served as a starting point but also highlighted the gaps in our understanding, since we were only basing our work off the original project description in PotentialProjects.pdf. After our meeting, where Dr. Kelly provided valuable insights into reinforcement learning and his research on Tangled Program Graphs (TPG), it became evident that our initial goals did not fully capture the project's scope and intentions.

Recognizing this, Richard reviewed Calvyn's draft, and together they realized the difficulty of defining meaningful goals without adequate context. This realization prompted a collaborative effort to refine and redefine our objectives. Over the course of the week, Calvyn and Richard synchronized their efforts, incorporating Dr. Kelly's feedback and the newfound understanding of the scope of Dr. Kelly's goals for our capstone project.

Through open communication and iterative refinement, we were able to resolve the initial ambiguity. This process not only clarified our goals but also strengthened our teamwork and alignment. The experience underscored the importance of seeking clarity from supervisors and maintaining continuous dialogue within the team to ensure everyone is on the same page.

By proactively addressing the lack of context and leveraging our collective efforts, we turned a significant pain point into an opportunity for growth and improved collaboration.

A second painpoint for this capstone project has been the lack of time to fully engage with the foundational aspects of the project itself. While crafting the proposal and development plan, the team largely feels like there has been little opportunity to thoroughly familiarize ourselves with the C++ library, experiment with new techniques, or test out different approaches that may help us define more conclusive statements in the problem statement and goals document. This time constraint has been particularly frustrating because building a strong technical foundation is critical for the success of the project, yet the tight deadline for the proposal has left limited space to explore and solidify these elements.

Due to this, the team may have to rely on estimation for certain details in the proposal, especially for the goals of the project. While these estimates are based on existing knowledge and given context, it certainly introduces uncertainty into the project planning process. The team believes the immediate need to meet proposal deadlines balanced with the longer-term need to fully grasp and develop the technical aspects of the project has been a major pain point throughout this phase.

3. **How did you and your team adjust the scope of your goals to ensure they are suitable for a Capstone project (not overly ambitious but also of appropriate complexity for a senior design project)?**

Adjusting the scope of our goals was a crucial step to ensure our Capstone project was both achievable and appropriately challenging. Initially, Dr. Kelly provided us with two primary objectives: code refactoring and interface development with MuJoCo. While these goals were clear, we recognized that their open-ended nature could lead to ambiguity if not scoped properly.

To address this, we decided to break down these overarching goals into smaller, more manageable subgoals. This decomposition allowed us to outline specific tasks, set realistic milestones, and better understand the steps required to achieve each objective. For example, under code refactoring, we identified the key components in a CI/CD pipeline, and for interface development, we listed the features that were essential for the project's success.

A significant factor in adjusting our scope was the need to fully comprehend the context of Dr. Kelly's research. Given that the scope of his work in machine learning and Tangled Program Graphs was outside our collective comfort zones, we initially found it challenging to gauge the complexity and time requirements of the tasks. None of us had prior experience in machine learning research, which made it difficult to estimate how hard each goal would be to accomplish.

To mitigate this uncertainty, we engaged in open discussions with Dr. Kelly to gain deeper insights into the project's demands. He reassured us that, with the right guidance, the challenges were very doable within our eight-month timeline. His confidence and willingness to provide support helped us feel more comfortable with the adjusted scope.