

# Personal Project: Computational Intelligence

## Objective A:

### Learning and Product Goal

#### Learning Goal

I was introduced to the idea of programming since I was a child since both of my parents work in the IT field. Because of this, I had an early, albeit simple education with programming, and in the nearly 5 years since I was first introduced to software, have developed a passion and skill around dealing with complex algorithms, in competing and studying for computing contests. Because of this, every time the news bursts out with a new huge advancement or increase in the field of Artificial Intelligence, I can identify where media has potentially inflated the conversation. However, I never fully understood how artificial intelligence worked. Therefore, when picking the goal for my personal project, I decided to start at the grassroots of modern artificial intelligence, in the study of artificial automata and computational intelligence. I will be motivated in this endeavor because of my previous passion.

#### Product Goal

The ultimate product will be to train a computational intelligence within the framework of a virtual game. The rules of the game will be set by me, and the goal is to create an algorithm that is most effective at scoring the most points. The only tools I can use will be the C++ language with the *std* library (as well as SQLite for file operations), because using somebody else's engine or creation takes the main portion of the project away.

The "game of life" for these algorithms will be played on a 20x20 board. Of these 400 squares, 80 of them will be filled with nutrients. Each algorithm will simply be a set of instructions based off the 5 adjacent squares, including the current one. Based on what these nine squares have in them, the algorithm will always make the same move in the same situation. The algorithm starts with 100 points. Every time it tries to run off the 20x20

board or it tries to consume nutrition that does not exist, it loses 10 points. Every time it consumes a piece of nutrition on its current square, it gains 5 points. Each game will consist of 300 steps, where each of them can either be moving or trying to consume nutrition. Below is an example of what part of the grid would look like. The A represents the algorithm's current position, the blue represents the field of view the algorithm considers, and the o's represent nutrition. Against an edge, the algorithm will sense a "wall unit".

Therefore, there are three possible square types for five positions, and each scenario can give 5 possible moves, either moving up, down, left, right, or consuming on the current square. Therefore, each algorithm can be saved as a  $3^5 = 243$  digit base-5 number.

In the situation below, the algorithm would be able to see that to their left, it is a wall, to their up and down are empty, and their right is nutrition. This would translate to a number  $i$  between 0 and 242, which would then correlate to a number  $m$  between 0-4 where  $m$  is the  $i^{\text{th}}$  digit of the base-5 number of the algorithm. Then we would translate that 0-4 number into one of the five possible moves.

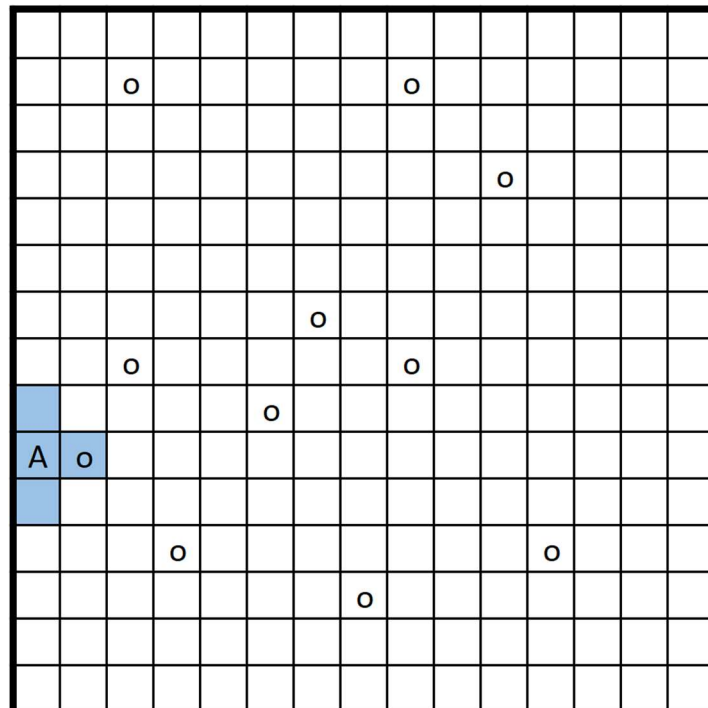


Figure -1: An example situation, where A Is the position of the individual, o are nutrition targets, and the vision is outlined in blue.

## Success Criterion

- The product will consistently train an algorithm(individual) that is able to score 200 points in one simulation. (Function)
  - o For points, reference the “Simulation Design” portion.
  - o The algorithm will be trained using a natural selection method, and this determines the effectiveness of my work.
  - o This will be tested by running the heuristic<sup>1</sup> function to test the final individuals between multiple trials of the product.
- The product will not be made with outside engines (Material)
  - o No external code or engine will be used to simulate natural selection.
  - o This is to ensure that I understand the theory behind computational intelligence.
  - o This will be tested by showing the entirety of the source code.
- The algorithm will be selected in 100 generations (Size)
  - o This is to ensure that the result is an effective algorithm.
  - o This will be tested by showing the results of all 100 generations.
- The algorithm will be stored as a string of numbers (Aesthetics)
  - o This is to ensure that the display and model is as minimal and compact as possible, so I can do it from scratch.
  - o This will be tested with pictures of a generation.

## Action Plan

In order to complete my project before the deadline, I will note down all the recommended times to complete different parts of the project in order to make sure that I am on track to finish before the final deadline. In addition, I will add to my calendar and schedule to make sure that I can pull out at least 2 hours a week (and more for most weeks) to work on the research and product portions of the project, including the writeup later.

Finally, it is important that all of the success criteria are fulfilled with the final product, therefore, every time I make an executive decision to

---

<sup>1</sup> A heuristic function tests the capability or result of a particular individual in the selection process.

modify a portion of the product, I will look back to my specifications to make sure that all of the success criteria are still fulfilled.

All these goals can be completed by using my self-management skills in order to effectively plan out my work and organize my work so that I can be as efficient as possible. A working action plan in accordance with the dates for advisor meetings and due dates has been included below, including the approximate time it will take to complete each task.

Date	Milestone	Notes		
10/12	First Adv. Meeting			<input checked="" type="checkbox"/>
Learning Phase				
	Determine game	Game found in App. A.	1 day	<input checked="" type="checkbox"/>
	SQLite (added 15/10/22)		1 week	<input checked="" type="checkbox"/>
	Natural Selection Algo.		1 week	<input checked="" type="checkbox"/>
	Mutation Algo.		1 week	<input checked="" type="checkbox"/>
11/23	Second Adv. Meeting			<input checked="" type="checkbox"/>
Product Phase				
	Create Population Function		1 week	<input checked="" type="checkbox"/>
	Create Heuristic Function		1 week	<input checked="" type="checkbox"/>
	Create Mutation Function		1 week	<input checked="" type="checkbox"/>
	Create Master Function		1 week	<input checked="" type="checkbox"/>
	Integrate SQLite for storage (added 15/10/22) (See Strand 2, OBJ 3)		1 week	<input type="checkbox"/>
	Debug and run trial		1 week	<input checked="" type="checkbox"/>
01/18	Third Adv. Meeting			<input checked="" type="checkbox"/>
01/27	Product Due			<input checked="" type="checkbox"/>
02/24	Writeup Due			<input checked="" type="checkbox"/>

## Objective B:

### ATL Skill 1: Research

The first of the ATL skills used in this project was Research. Starting from almost scratch, I would need a deep understanding of both natural selection algorithms as well as general programming practices and basic algorithms and data structures in order to develop clear, concise, and efficient code. Most of the research would happen within the learning stage of the process, and only minimal google searches to clarify would be necessary during the product stage.

Using a combination of online and physical resources, I took notes around my topic, and developed a deeper understanding of the method as well as learned different algorithms and data structures that would help me make my code more efficient and better organized.

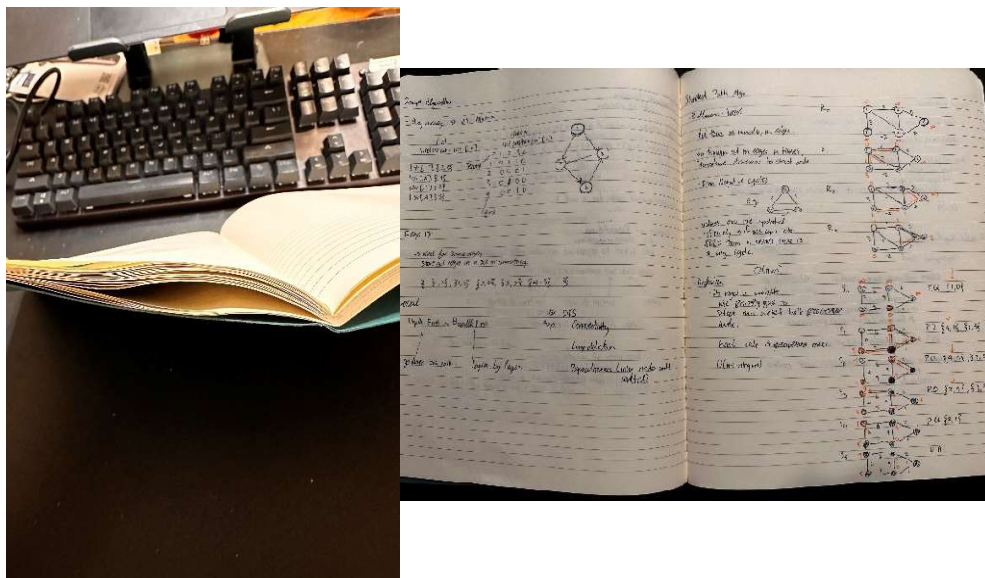


Figure 0 - Notes from Guide to Competitive Programming - Antti Laaksonen

In specifics, I used media literacy to determine the usefulness of the methods and structures in the book to keep track of all the possible solutions I could have to my specific project, by organizing all the algorithms that I had learned into different usages and by order of

efficiency, as seen in figure 1. In addition to that, I always checked methods mentioned in the book to make sure that they were up to date, as well as to see how else they can be adapted to be used and included this research into my original notes.

Graph Search	Other Graph	DP	Other
DFS	Kruskal's	Knapsack	Greedy
BFS	Prim	2D	
Dijkstra	Maximum Flow	LIS	

Figure 1 - Part of the Classification of algorithms

Also given the tight time limit of the project, I couldn't fully dive into the specifics and extensions of all the algorithms mentioned in the book, so I had to field the initial description and explanation of all of them and make informed choices into which ones seemed the most promising for my circumstance, and to explore those in depth.

The second most important source for me was around the actual algorithm that was to be used. In the end, I chose to develop a natural selection algorithm to mimic the natural selection that happens as a part of evolution in order to develop entities or beings that could effectively find food on a grid (Mitchell)<sup>2</sup>. This source in combination with many other sources from various medium were used, in order to get a wide range of opinions and ideas about how to best proceed in terms of the algorithm. I practiced media literacy by trying to understand why each source had a different point of view and synthesized the information in order to find my own decision that would benefit me best. This was immensely important afterwards, in the product period, because finding the simplest yet effective algorithm allowed me to save a lot of time and complete the project in a timelier manner.

---

<sup>2</sup> The Book deals with information theory. The chapter *Life and Evolution in Computers* talks about self-generation and natural selection algorithms.

In terms of information literacy, it was important for me to connect various sources and ideas together that were presented in the text to be able to field a wide view of the possibilities presented when using many different methods or structures in tandem with each other. Not only was it important for me to keep good notes and diagrams to allow me to develop better long-term memory (Figure 1R, an example of diagrams for shortest-path algorithms in a graph), but it was also important to collect a variety of ideas from across the spectrum in terms of algorithms in order to develop a full picture (figure 1L, the number of notes taken). In addition, I wrote an OPVL regarding the source *Guide to Competitive Programming*, the following is an excerpt from the limitations section of the OPVL (Laaksonen).

*“The source only included general algorithms that are used for competitive programming, which has limited probability of being useful for my final product. Only some of what I learned from the book can be used. In addition, most of the information was designed for competitions, to be quick and easy to implement, which isn’t a necessity for me, and more complex, but efficient methods can be used.”*

*-Excerpt from OPVL of Guide to Competitive Programming*

The OPVL was important to me because it allowed me to organize my thoughts about how to use the information that I retrieved from the source. Doing the OPVL allowed me to realize the limitations and values of the source and allowed me to effectively understand what I knew still needed to know, which allowed me to find out and fix leaks in my understanding.

Synthesizing these two main sources was also very important to the success of my project. Being able to connect the general algorithms that I learned from the competitive computing book and the specific algorithms from the genetic algorithm source. Being able to combine these two is the core idea of my project, so being able to organize and connect these two main particles of knowledge makes me able to quickly finish my project, since most of the thinking around how to implement my ideas will be already dealt with.

## ATL Skill 2: Self-Management

Self-management was an important part of my project, because the entire codebase for my project was larger than I had ever handled before. In competitive programming, which was what I was used to, programs would almost never pass 100 lines, but this program almost hit 300, 3 times more than I was used to. Therefore, for me, being able to keep track of what I needed to mend or work on was vital to progress efficiently in my project. The first most important part of the project were my organizational skills, because I had to carefully plan and execute my thoughts. Second to that were my affective skills, so that I could keep positive and committed through the journey.

Firstly, I created a Kanban<sup>3</sup> board using an organizational program called *Notion* to track my project as it was going and used that to track what I needed to do.

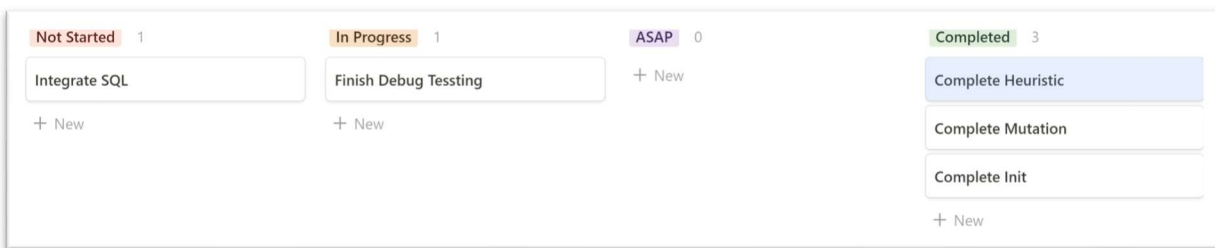


Figure 2 – Kanban board for my project

Additionally, in my calendar, I planned times where I would work between my schedule of extra-curriculars and classes, so that I would find time to finish the project and meet deadlines. It was important for me also to balance the use of technology, since I used a lot of technology to organize and plan my time, it was imperative that the technology did not distract me during the time that I was working. Therefore, I turned off my notifications on my phone and set my headphones to noise cancellation, so that I could focus on my work.

---

<sup>3</sup>A Kanban board is an organizational tool where different tasks are moved into different sections, by urgency, completion, and importance.



My calendar also included deadlines that I set for myself so that I would stay on time for the finishing of the project.

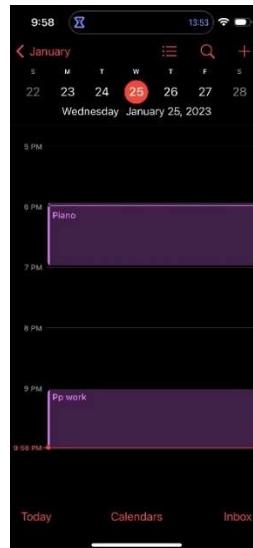


Figure 3 - A Sample calendar planned session.

Also, during and before starting on the project portion, I would draw rough ideas of how the different parts of code would be interconnected, such as the one in figure 3. These helped me immensely, because not only did they help me plan out the different portions of the project so that I didn't forget anything, but they also helped me be more efficient with my time, since I was able to visualise easier, and always had something to go back to if I lost my train of thought or was otherwise distracted.

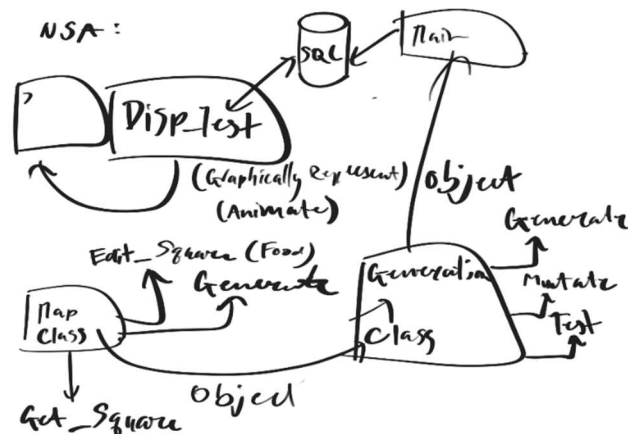


Figure 4 – Rough sketch of the structure of the program

In terms of affective skills, it was also important for me to continue to be excited and focused when working on the personal project. Therefore, in order to deal with boredom, loss of interest, and therefore failure, I started by creating a simple to-do list with easy and small tasks.

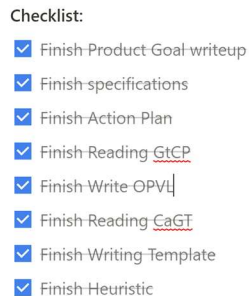


Figure 5 – To-Do checklist (small tasks have been removed over time)

These small ticks where really simple for me to check off even trivial tasks, which made staying focussed and excited for the project easier. In addition, it also helped me with self-motivation, because getting checkmarks can almost be addicting, and once you start working and finishing tasks, it's very hard to stop.

Resilience was also a big part of the project. Almost always, when you first finish a large project of code, there will inevitably be a lot of bugs, or issues with the logic. Therefore, debugging was one of my main tasks in the project, and sometimes, it can feel like there's an overwhelming amount of problems to solve. To combat this, I split the project into many parts to debug, so that I wouldn't be exposed to all of the problems at once, which would cause me to lose heart. This proved to be very successful for me, since I was able to complete the task in a efficient and timely manner, and that allowed me to have more buffer time at the end for the report writing as well as other smaller tasks.

## Objective C:

### Impact on Me

```
class board {
private:
    char grid[20][20]; //0 is empty, 1 is food
public:
    void setboard(){
        //reset board
        memset(grid, 0, sizeof(grid));
        srand((unsigned) time(NULL));
        for(int i = 0; i < 20; i++){
            for(int j = 0; j < 20; j++){
                int rng = rand()%100;
                if(rng > FOODPERCENT) grid[i][j] = 1;
            }
        }
    }
    int get(int r, int c){
        // get data from square (r, c)
        if(r>=0 && r<20 && c>=0 && c<20){
            return grid[r][c];
        }
        else return 2;
    }
    int eat(int r, int c){
        if(grid[r][c] == true){
            grid[r][c] = false;
            return true;
        }
        else return false;
    }
};
```

Figure 6 – The *class* structure used to organize the playing board.

This project gave me a good opportunity to not only practice skills that I had previously had with programming, but also complete my learning goal and work with larger projects and code. Being able to command these skills fluently will be important to me in the future since I wish to go into information technology and software.

Additionally, the learning process of my project allowed me to explore into the specifics and algorithms behind automata and computational intelligence.

These topics are now at the forefront of technology, with AI and Machine Learning penetrating the everyday of our lives, understanding and being able to create even the most basic forms of these technologies will allow me to explore careers and work that will interest me in the future.

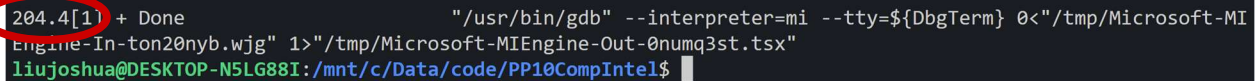
Also, the learning process allowed me to understand the algorithms and methods that I used more clearly, which I can apply today in my other projects and competitions. I think that these two

takeaways from the learning process alone makes the project worthwhile, because almost all the information that I collected is useful in the near future, if not immediately. Effectively implementing my research skills helped me to develop as a learner, as I got more practical experience finding specific and general material to supplement each other. This allowed me to grow as a learner, since I improve the efficiency of my research and synthesis of information, as well as improving my note-taking skills.

Lastly, the actual product creation process was a challenging and creative process for me, since it included using *class* structures (such as the one in figure 6), which was the main feature of the language I chose (C++) that I did not have practice with before. Being able to practice and learn new skills and features that I previously didn't have experience with was the main takeaway for me in the product phase. Not much of the coding part was immensely complicated or complex, but the size and nature of the project forced me to use methods that I didn't need to before. Although being a challenging task, being able to undertake it and finish a complex project using only simple methods felt like a great accomplishment to me. Self-management skills was a huge part, as I had many different pieces of the project that had to work together, where if one part changed, the rest had to follow. Organizing my work in different portions allowed me to keep track of my changes and make sure every part was on the same version.

## Evaluating the Success of my Project

**The Product will consistently score 200 points.**

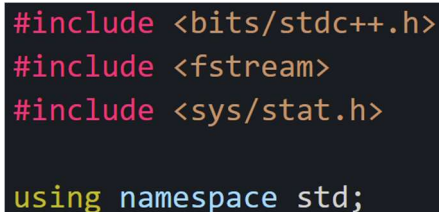
A terminal window with a dark background. The first line shows the output "204.4[1] + Done" where "204.4[1]" is circled in red. The second line shows a command: `"/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-ton20nyb.wjg" 1>"/tmp/Microsoft-MIEngine-Out-0numq3st.tsx"`. The third line shows the prompt `liujoshua@DESKTOP-N5LG88I:/mnt/c/Data/code/PP10CompIntel$` followed by a cursor.

```
204.4[1] + Done
"/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-ton20nyb.wjg" 1>"/tmp/Microsoft-MIEngine-Out-0numq3st.tsx"
liujoshua@DESKTOP-N5LG88I:/mnt/c/Data/code/PP10CompIntel$
```

Figure 7 - Output of a program to get the average score of final generations in different trials.

As seen in figure 6, the final generations of 5 trials averages out to 204.4 points. Therefore, this specification has been achieved.

**The Product will not be made with outside engines.**

A code snippet on a dark background showing the beginning of a C++ program. It includes three `#include` statements for `<bits/stdc++.h>`, `<fstream>`, and `<sys/stat.h>`, followed by a `using namespace std;` statement.

```
#include <bits/stdc++.h>
#include <fstream>
#include <sys/stat.h>

using namespace std;
```

Figure 8 - The beginning of the program, where all the libraries used are declared.

As can be seen in the above picture, the entirety of the libraries used are declared. Nothing outside of system libraries, file libraries, and the basic standard C++ library was used. Therefore, the entirety of my project runs without the use of outside code, and the entirety of the working project is within this one C++ source code file.

**The algorithm will be selected in 100 generations.**

```
const long int GENERATIONCNT = 100;
const long int TESTCNT = 50;
const long int GENERATIONSIZE = 100;
const long int FOODPERCENT = 40;
const long int MOVES = 400;
```

Figure 9 - The beginning of the program, with the constant variables

In figure 8, we can see that in the constant values for the project, the value GENERATIONCNT is set to 100. This means that the results of the program will be determined in 100 generations of the project iteration.

**The algorithm will be stored as a string of numbers.**

```
40002332241004414301014223202100133232404223243244102423113201023024203010233124220143012002100133
13320001130322323041031031321343030024131023044144040321110441414243314430211404040402001434034440
4001442042033330321231011310430444310223003344440132101224230424132431202102413413410244330241201
```

Figure 10 - An excerpt from the files containing the generation data.

The algorithm stores all the generation units as files with the strings of numbers representing each individual. Although to humans, it is very difficult to understand what each algorithm really means, but for a computer, this format is really simple and efficient in terms of mutating and testing.

### **General Overview:**

In general, all the specifications for my design were met. Although I hoped to have been able to achieve a higher score, perhaps it was because the number of generations was not sufficient to create individuals capable of scoring higher than around 200. If I ever do a similar product, I will be sure to test with varying levels of generations, in order to see whether if the time it takes to develop has an impact on the success of the final generation (More so how much effect it has).