Timeline to Final Review

Mostafa Rizk

January 8, 2021

Following the recommendations of the panel, and in discussion with my supervisors, I have streamlined my research to focus on 3 research questions. The timeline below (Section 5) consists of 4 groups of tasks, one for each research question and one for the literature review, which is ongoing parallel to the other 3 groups. The timeline goes from the last 2 work weeks of 2020 through to the reconvene seminar in April. Work after the seminar will be primarily thesis writeup. Sections 1, 2, 3 and 4 explain each group of tasks. Section A in the appendix contains a clarification of the learning process to aid in understanding some of the tasks.

1 Group 1- Landscape Analysis

For this group of tasks, I perform RWG analysis, as described in [1] to understand the fitness landscape for the centralised setup. This involves randomly sampling a team of agents (specifically 2 agents) several thousand times, evaluating the team and visualising the distribution of team scores. I can then make inferences about things like the gradient of the landscape and the presence of local optima.

1.1 Creating a Fitness Gradient

One possible shortcoming of this part of the research is that the fitness function might be making the problem unnecessarily difficult.

Hypothesis 1.1: Rewarding agents for partial retrieval of the resource will create a smoother fitness gradient that evolution could then take advantage of.

Investigating this hypothesis is the first task in the group. I conducted an experiment to test this hypothesis where I repeated the landscape analysis but with incremental rewards. The results of this experiment are in Section B of the appendix.

1.2 Visualising Specialisation

I already performed the initial analysis of the landscape, however, the visualisation did not contain any information about how specialised each team was i.e. was a given solution a pair of generalists or a pair of specialists? Are the better solutions in the landscape usually specialists? How common are specialist solutions compared to generalist ones? To visualise this information I need a way of measuring the degree of specialisation of a solution, so the second task in this group is to evaluate some candidate metrics and the third is to incorporate the chosen metric into the landscape analysis.

Hypothesis 1.2: The fittest solutions in the landscape are the specialised ones but they are also rarer than generalist solutions.

1.3 Dimensionality Reduction

I have also included dimensionality reduction as part of the third task. Following the seminar, I discovered a technique to project the fitness landscape in two dimensions, which lets me visualise how far apart solutions are. This technique, Sammon's mapping [2], has been used in other work [3, 4] to visualise landscapes for agent-based problems. Combined with the specialisation metric, this answers additional questions that are also significant. How far apart are specialist and generalist solutions in the landscape? How many optima are there? What trajectory does evolution take through the landscape? I have already implemented this technique and started to get preliminary answers.

Hypothesis 1.3: There are two optima in the landscape, a specialist one and a generalist one, and the specialist optima has fitter solutions.

1.4 Analysis of Different Slope Angles

The fourth task is to redo the analysis using different slope angles. This means randomly sampling teams of agents as before, but evaluating them in environments where the slope is different. Changing the slope angle changes how beneficial specialisation is. Having no slope at all means there should be no benefit from specialisation and no pressure for agents to learn it.

Hypothesis 1.4: When the slope is flat, no solutions in the landscape will be specialists and when the slope is steep, the specialised solutions will be higher-scoring.

1.5 Optional Tasks

The two optional tasks are to get a more thorough understanding of the landscape, if time allows, by considering the landscape for the decentralised setup and for a more complex version of the environment.

1.5.1 Decentralised Setup Analysis

The current landscape analysis shows the landscape for a centralised algorithm because it plots solutions for the team as a whole. For the decentralised algorithm, the fitness landscape is not static because an agent's fitness depends on who it is paired with. The features of the landscape, therefore, are likely to be different for the decentralised approach. This task involves modifying the current landscape analysis to give insight into the landscape for the decentralised setup. One way of doing this is by using the dimensionality reduction technique from Section 1.2. Consider 2-agent teams in the centralised setup. A solution in the landscape is the genomes of both agents on the team. Consider now a 2-agent team in the decentralised setup. Each agent has its own fitness landscape, but if we take both agents on the team and concatenate their genomes, we can plot that team in the same landscape as the centralised teams. This does not tell us what the landscape is like for each agent during the learning process, but it shows us how the decentralised algorithm navigates the shared fitness landscape compared to the centralised algorithm. Does the decentralised algorithm ultimately find the same solutions as the centralised one?

Hypothesis 1.5: Solutions found by the decentralised algorithm will be primarily generalist whereas solutions found by the centralised algorithm will be primarily specialist.

1.5.2 Analysing Complex Environments

The current task is relatively simple. How does the landscape change when the task becomes more challenging?

2 Group 2- Centralised vs Decentralised

Following from the description of the learning process in Appendix Section A :

Hypothesis 2.1: Centralised learning finds specialisation more frequently.

Since centralised learning is trying to find the best team solution and specialisation is what is best for the team, it should more frequently find specialisation than decentralised learning. In decentralised learning, each agent's learning process is trying to find the best solution in an environment where it does not know who it will be partnered with. The best solution should thus be the safest one, which is a generalist behaviour.

Hypothesis 2.2: Decentralised learning scales better than centralised learning.

For centralised learning, since it is searching for a team solution, the search space grows rapidly as the number of agents increases, so we can expect learning to find worse solutions for larger teams. These solutions should be less fit but it is not clear if they will also be less specialised.

For decentralised learning, the space of solutions remains the same for each agent, when the team size increases, however, there are more concurrent learning processes. An agent's score depends on many other agents executing unknown strategies in its environment, and all those strategies affect one another, so the increased noise may also make it difficult to find good solutions for larger teams.

Hypothesis 2.3: Decentralised learning is more robust to changes in the team (as measured by the fitness of solutions).

As stated in Hypothesis 2.1, decentralised learning should find the safest strategy for an agent, i.e. a generalist strategy. This strategy is successful regardless of the strategies of its teammates. If an agent is evaluated with new teammates it has never been paired with before, it should still perform reasonably well.

Conversely, an agent from the centralised setup is part of a larger solution. It is likely specialised. We can expect that if the agent is paired with new teammates, the team performance will suffer.

The first task in this group is to do some code fixes to facilitate the experiments that will answer the hypotheses. The second task answers Hypothesis 2.2 and the third answers Hypothesis 2.3, while both together answer Hypothesis 2.1 as specialisation can be measured for the solutions in both.

3 Group 3- Price of Anarchy

Theoretically, centralised learning is equivalent to a benevolent dictator that assigns roles to each agent on the team so that the collective solution is socially optimal. Decentralised learning is equivalent to a group of self-interested individuals where each is trying to find what is best for itself and we expect that is less likely to find a socially optimal solution. Calculating the price of anarchy lets us determine how far the team performance drops when agents are self-interested, i.e. how much performance should drop when using decentralised as opposed to centralised learning.

Hypothesis 3.1: The slope foraging problem is a stag hunt with two Nash equilibria, where the payoff-dominant equilibrium is specialist solutions and the risk-dominant is generalist solutions.

Hypothesis 3.2: The centralised setup is more likely to find the payoff-dominant Nash equilibrum while the decentralised setup is more likely to find the risk-dominant Nash equilibrium.

4 Group 4- Literature Review

I have created a spreadsheet to systematically keep track of the literature and ensure my research is supported by a firm knowledge of the field. Throughout the next few months, I will add relevant papers to the spreadsheet as I delve deeper into each research question, doing brief first passes of each to get an overview and deeper second and third passes for any that are especially relevant.

5 Timeline

Roadmap to Pre-submission

Mon, 12/7/2020

1

TASK	PROGRESS	START	END	DURATION	07-Dec	14-Dec	21-Dec	28-Dec	04-Jan	11-Jan	18-Jan	25-Jan	01-Feb	08-Feb	15-Feb	22-Feb	01-Mar	08-Mar	15-Mar	22-Mar	29-Mar	05-Apr	12-Apr
What are the fundamental features of the fitness landscape of a task specialisation problem?																							🟩
Investigate landscape smoothing	100%	4-Jan	8-Jan	1w					▨														🟩
Implement 1-3 metrics for specialisation and choose the best one	0%	11-Jan	15-Jan	1w						🟩													🟩
Redo landscape analysis with specialisation metric and dimensionality reduction	0%	18-Jan	22-Jan	1w							🟩												🟩
Perform RWG analysis with different slope angles	0%	25-Jan	31-Jan	1w								🟩											🟩
Create analog of landscape analysis for decentralised setup (optional)	0%	3-Feb	14-Feb	2w									🟩	🟩									🟩
Perform landscape analysis for a more complex version of the environment (optional)	0%	17-Feb	28-Feb	2w											🟩	🟩							🟩
When do decentralised solutions outperform centralised solutions in task specialisation problems?																							🟩
Implement code fixes (seeding, cooperative coevolution, multithreading)	80%	7-Dec	29-Jan	8w	▨	▨			▨	▨	▨	🟪											🟩
Re-run scalability experiments (varying team size)	0%	1-Feb	12-Feb	2w									🟪	🟪									🟩
Run robustness experiments (change agents on the team)	0%	15-Feb	26-Feb	2w											🟪	🟪							🟩
What is the price of anarchy in task specialisation problems?																							🟩
Formulate a game	0%	1-Mar	19-Mar	3w													🟧	🟧	🟧				🟩
Find nash equilibria	0%	22-Mar	9-Apr	3w																🟧	🟧	🟧	🟩
Study learning dynamics	0%	22-Mar	9-Apr	3w																🟧	🟧	🟧	🟩
Perform synthesis on basis of the data	0%	22-Mar	9-Apr	3w																🟧	🟧	🟧	🟩
Review literature																							🟩
Ongoing review of literature	10%	7-Dec	11-Apr	16w	▨	▨			🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟩

References

[1] D. Oller, T. Glasmachers, and G. Cuccu, "Analyzing reinforcement learning benchmarks with random weight guessing," *arXiv preprint arXiv:2004.07707*, 2020.

[2] J. W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Transactions on computers*, vol. 100, no. 5, pp. 401–409, 1969.

[3] Y.-H. Kim and B.-R. Moon, "New usage of sammon's mapping for genetic visualization," in *Genetic and Evolutionary Computation Conference*, pp. 1136–1147, Springer, 2003.

[4] S. Risi, C. E. Hughes, and K. O. Stanley, "Evolving plastic neural networks with novelty search," *Adaptive Behavior*, vol. 18, no. 6, pp. 470–491, 2010.

[5] F. Silva, M. Duarte, L. Correia, S. M. Oliveira, and A. L. Christensen, "Open issues in evolutionary robotics," *Evolutionary computation*, vol. 24, no. 2, pp. 205–236, 2016.

[6] Y. Wei, M. Hiraga, K. Ohkura, and Z. Car, "Autonomous task allocation by artificial evolution for robotic swarms in complex tasks," *Artificial Life and Robotics*, vol. 24, no. 1, pp. 127–134, 2019.

A The Learning Process

Each agent is controlled by its own neural network. The network encodes the agent's behaviour i.e. it encodes a function that maps from local observations to actions. The neural network architecture is fixed, but the weights are changed throughout the learning process.

A.1 Centralised

In the centralised setup, a learning algorithm creates a population of solutions. A solution is a sequence of neural network weights for the network of each agent on the team. For example, if a network has 50 weights and there are 4 agents, the solution has a length of 200. Each part of the solution is assigned to a random agent. The agents then carry out several episodes, starting from random positions at the nest each episode and acting for a fixed number of time steps. The score is calculated at the end of each episode and it is the sum of rewards for all resources gathered by the team minus the costs paid by all agents. The average score over all episodes is calculated and used to create a new population of solutions. This repeats until the algorithm terminates.

A.2 Decentralised

The decentralised setup has changed since the seminar and now uses a technique called cooperative coevolution. In cooperative coevolution, rather than one evolutionary process learning a combined solution for the entire team, each agent on the team has its own evolutionary process learning a solution for only itself. For example, if there are 4 agents, there are 4 evolutionary processes running concurrently. At generation i, each of the 4 processes creates a population of solutions for its agent, where a solution is the weights of a single agent's neural network. The score of a solution at generation i is calculated by pairing it with the best solutions from each of the other 3 agents' evolutionary processes at generation $i-1$. The score is for that single agent's performance; it is the team reward (total resources collected) divided by the number of agents (regardless of their contribution) minus the cost incurred by that particular agent.

A.3 Difference

The difference between centralised and decentralised is that a centralised algorithm uses one learning process to learn a solution for the entire team whereas the decentralised algorithm uses multiple concurrent learning processes to learn solutions for each agent independently that are then combined. Centralised learning rewards the team while decentralised learning rewards the individual.

B Creating a Fitness Gradient

Hypothesis 1.1: Rewarding agents for partial retrieval of the resource will create a smoother fitness gradient that evolution could then take advantage of.

I divided the reward for retrieving a resource by the length of the arena and gave agents part of the reward if they moved the resource a single tile closer to the goal. The plots for the original landscape analysis and the analysis with the new fitness function are shown in Figure 1 and Figure 2 respectively.

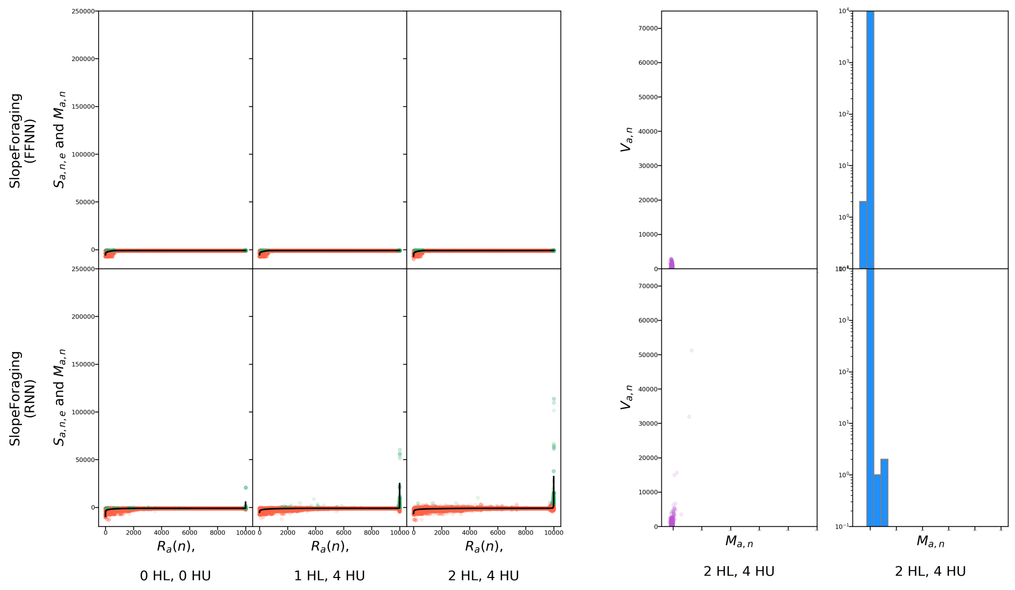

Figure 1: RWG Analysis with the original fitness function

Looking at the 6 mean plots in Figure 2 compared to Figure 1, we see that with incremental rewards, the solutions do in fact form more of a gradient towards the right end of the plot, in particular for RNNs (bottom row). This means that a lot of the solutions in the landscape are partially successful at retrieving resources. However, the landscape is still largely flat. This is presumably because most solutions in the landscape don't even get to the point of picking up a resource from the source.

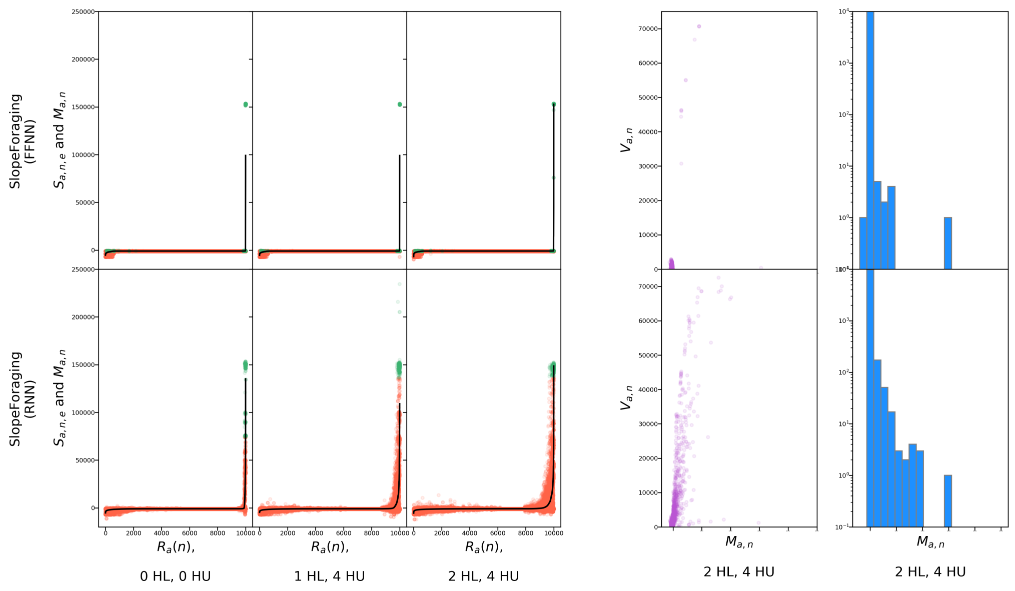

Figure 2: RWG Analysis with incremental rewards

Thus the answer to Hypothesis 1.1 is:

Answer 1.1: Rewarding agents for partial retrieval of the resource creates a smoother fitness gradient, but the landscape is still largely flat.

In order to further smooth the fitness landscape there would need to be rewards for partial progress up the slope, but the problem with this is that it is likely to disincentivise the collector behaviour in evolution. Why would an agent stay at the nest and collect resource when they are rewarded for going up the slope? Additionally, while this problem is simple enough to smooth the fitness gradient in this way, such smoothing may not be possible for other more complex problems. The purpose of this research is to understand how to use AI to find good solutions with minimal (if any) human intervention.

Moreover, the flatness of the landscape is, to some extent, inherent to this task. This is a problem common in evolutionary computation known as the bootstrapping problem [5, 6]. Bootstrapping is "when the task is too demanding for the fitness function to apply any meaningful selection pressure on a randomly generated population of initial candidate solutions" [5]. The bootstrapping problem often occurs when the goal behaviour is complex relative to the very simple available actions [6]. For this problem, the available actions are primitives like 'move forward one tile' and 'move backward one tile'. When putting together sequences of these actions, most sequences will obviously not be very successful. There are many proposed solutions to solving the bootstrapping problem and they fall under the broad categories of inserting human knowledge into the learning process or increasing the diversity of solutions [5]. Smoothing the landscape falls under the former category, but this family of techniques has some shortcomings, in particular it reduces the potential for automation and risks the experimenter introducing negative biases. Using rwg to find a seed falls under the second category; it has the shortcoming of additional computation, which may not scale well for problems with larger solution spaces, but for those problems there is a wealth of alternative techniques for increasing diversity, such as novelty search. In keeping with the spirit of AI, I have chosen to continue using the more challenging fitness function.