Final Report: Neuroevolution of Task Specialisaiton for Multi-Agent Systems

Mostafa Rizk

March 15, 2021

1 Introduction

2 Landscape Analysis

What are the fundamental features of the fitness landscape of a task specialisation problem?

2.1 Visualising Specialisation in the Landscape

2.1.1 The Landscape

To understand the landscape, I use RWG Analysis as proposed by [1]. This approach is relevant to my problem as it is intended for the analysis of agent-based reinforcement learning problems that use neural networks. Oller et al use the RWG (random weight guessing) algorithm which works by randomly sampling solutions from a distribution. A neural network-based agent that solves a task can be represented as a string of weights. Similarly, a team can be represented by a string of weights for each team member. Every such string is a solution (or genome). We can obtain a random solution by sampling from a distribution. If a large enough number of solutions are sampled (several thousand) and each one is evaluated for several episodes, we can gain an understanding of how difficult the problem is by visualising the distribution. By comparing different neural network architectures, we can also study

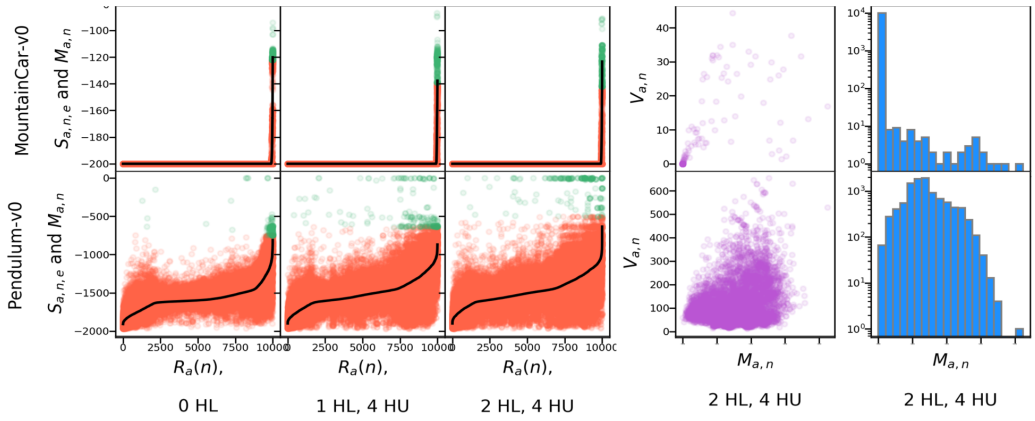

Figure 1: RWG analysis of OpenAI Gym benchmarks (reproduced from [1])

how the architecture affects the problem difficulty. While not a direct observation of the fitness landscape, it gives insight into the difficulty, modality and presence of plateaus by showing how common good solutions are and whether or not the sorted solutions have a gradient of improvement. Oller et al apply their technique to known problems in the reinforcement learning domain, making it possible to contextualise the SlopeForaging problem in relation to those problems.

In Fig 1, each row represents a known problem, selected from the OpenAI Gym Benchmarks. The first three columns show different neural network architectures used by the agent (feed-forward neural networks with 0, 1 and 2 hidden layers, respectively). In each of those columns, the plot shows 10,000 randomly sampled solutions, each dot representing 1 of 20 episodes for that sample, sorted in order of their score, showing that for some problems, like Pendulum, solutions can be incrementally improved, meaning there is a gradient that a search algorithm can follow. However, for other problems such as MountainCar, most solutions perform equally poorly with only a few being successful. For this class of problem, Oller et al suggest using algorithms that are highly exploratory since there is no gradient to follow. The last two columns show the variance of the sorted solutions and the number of solutions that fall into each score range. For example, for MountainCar, there is low variance for low-scoring solutions and most solutions fall into the leftmost bucket, corresponding to low scores.

2.1.2 Specialisation

However, one shortcoming of RWG analysis is that it was not give insight into the degree of specialisation of the discovered solutions. Is a given solution a pair of generalists or a pair of specialists? Are the better solutions in the landscape usually specialists? To visualise this information I first need a way of measuring the degree of specialisation of a solution, I can then overlay this on the landscape plots.

To quantitatively measure specialisation, it is important to first define what specialisation is. For the purpose of this research, specialisation is when a team of agents cooperate, by performing different roles, to accomplish the task more effectively than a team of generalist (i.e. independent) agents. Agents cooperating by doing the same role, e.g. carrying a resource together, is not considered specialisation. Agents performing different roles that do not provide an advantage over a generalist team (e.g. one agent doing a generalist strategy and another agent being stationary) is also not considered specialisation.

This research is only concerned with tasks where specialisation improves performance and how to effectively evolve specialisation for those tasks. In real-world applications, we may not know a priori what the specialist solution to the problem is, but for the slope foraging problem, we know that it involves a subset of agents dropping resources from the top of the slope and the rest of the agents collecting the dropped resources from the cache. Therefore, our metric must be one that indicates these dropper and collector behaviours are being enacted to retrieve resources.

The way specialisation is measured in the literature uses one or more of the following:

Strategy switches

Nitschke et al [2] count times strategy was switched over all possible switches. Less switches means more specialised. Gautrais et al [3] do the same but take the complement. Gigliotta et al [4] counts time spent changing strategies.

Pini et al [5, 6] use an analog of Ferrante et al's slope foraging [7], where there is a source and a nest and agents can either take a long corridor to the nest or place items in a cache that transports them to the other side, where their team-mates take them to the nest. They count the number of times the cache was chosen instead of the corridor as an indicator of the degree of specialisation.

These are not easily applicable to this slope foraging implementation because it uses low-level actions, whereas they assume agents can directly choose a high-level strategy. Agents changing from "go forward" to "go backward" doesn't tell much about specialisation. But if agent actions were just "do dropper behaviour", "do collector behaviour" and "do generalist behaviour", it would be more feasible to use measures like these works. Some measures also look at specialisation of the individual rather than the team, which is not as meaningful for slope foraging.

Diversity and Entropy

Li et al [8] group agent genotypes into clusters and use Shannon's entropy to measure the diversity of the team. Each cluster is believed to be a specialist strategy. Specialisation is defined as "the part of diversity that is demanded for better performance" so a diverse team is only specialised if their diversity is correlated to improved performance. Li et al then look at diversity and performance through the learning process and calculate their correlation coefficient as a measure for the degree of specialisation of the final strategy(?)

This is computationally expensive because it requires data on team performance throughout all of the learning process and requires integration over all algorithm configurations (?). Additionally, since the authors use a PFSM, genotypic diversity implies phenotypic diversity (?). Using a neural network, different genotypes might still have the same behaviour. Since then, work has been done on diversity-based search algorithms. That literature uses behaviour characterisation to describe the behaviour being done. Agents could be clustered according to this, but diversity does not necessarily indicate specialisation or cooperation. More recent work by Gomes et al [9] use coevolution to evolve heterogeneous cooperative teams, they count the number of different agents or populations. Populations with similar behaviours are merged so all agents on the team are different, but again, this does not nec-

essarily indicate that the team has specialised.

O'Donnell and Jeanne [10] calculate the entropy of an individual agent, specifically the proportions of its time spent doing different activities (i.e. the diversity of resources it gathers). They use it directly as the degree of specialisation. This runs into the same problem as before, where the metric is difficult to apply to slope foraging because agents do not choose high-level strategies. In addition the metric is for individuals and does not apply to this context because agents only gather one type of resource. Balch?

Resource retrieval

Ferrante et al [7] have a comparatively simple metric. They count the proportion of resources retrieved cooperatively over the total resources retrieved. This is independent of whether actions are high or low level and doesn't need the full evolutionary history. It is designed for slope foraging so is appropriate, though it has shortcomings. Cooperative retrieval simply means a resource was picked up by two or more agents before being delivered. An agent could pick up and drop the resource at the source and a generalist could come to collect it and that would count as a specialist strategy. Technically, this is still a dropper-collector strategy, but it does not capture the type of dropper-collector strategy that improves performance.

Additionally, it does not account for 'hitch-hikers' i.e. agents doing nothing while the rest of the team retrieves all resources cooperatively. Since the Ferrante setup is homogeneous (i.e. all agents have the same policy), it is unlikely that some would cooperate while others hitch-hike, but since this research considers heterogeneous teams, hitch-hikers are more likely to be encountered, so the specialisation metric should distinguish between teams that have a hitch-hiker and teams that don't. It also can't distinguish teams where a dropper and collector swap roles part-way. This is unlikely to occur, though. In short, the Ferrante metric's main shortcoming is that it does not distinguish *imperfect* specialists.

2.1.3 Choosing a Metric

Based on a review of techniques in the literature, the metric used by [7] is the most appropriate. However, to overcome its shortcomings, I propose to

add two new terms to the formula. Ferrante et al measure a term I will refer to as the *cooperation rate* or:

$$R_{coop} = \frac{N_{coop}}{N_{total}}$$

where:

N_{coop} is The number of resources retrieved cooperatively (i.e. picked up by at least 2 agents before being delivered to the nest)

N_{total} is The total number of resources retrieved by the team

To account for instances of imperfect specialisation where agents cooperatively retrieve a resource but drop or collect it in the wrong place, I introduce the *efficient cooperation rate*:

$$R'_{coop} = \frac{N'_{coop}}{N_{total}}$$

where:

N_{spec} is The number of resources retrieved using efficient cooperation (i.e. the resource was dropped on the slope, picked up from the cache and the agents that dropped and picked up were different)

To capture the spectrum of specialisation, I take the average of the two terms:

$$R_{spec} = \frac{R_{coop} + R'_{coop}}{2}$$

To detect "hitch-hikers", I also introduce the *agent participation*:

$$P = \frac{A_{active}}{A_{total}}$$

where:

A_{active} is The number of agents that participated in the retrieval of at least one resource

A_{total} is The total number of agents on the team

The final measure of degree of specialisation is then:

$$S = R_{spec} \times P$$

One shortcoming of this metric is that it does not detect role-switching. For example, a team composed of a dropper and a collector but halfway through, the dropper becomes the collector and vice versa. Role-switching can be detected by observing the variance of an agent's y-coordinate in the arena. A dropper or collector that remains in their role throughout runtime will move at most 1 tile in either y direction, so they will have a low variance. An dropper that changes roles will have a high variance, akin to a generalist. The average y-variance of a team indicates role-switching and, to an extent, specialisation. The average variance could potentially be used to measure specialisation on its own, though it would not detect hitch-hikers or even just teams that are completely stationary. For this reason, and to avoid over-complicating the metric, I only use the variance as an additional measure to check for role-switching.

2.1.4 Assessing the metric

Team Strategy	$\boldsymbol{R_{coop}}$	$\boldsymbol{R'_{coop}}$	$\boldsymbol{R_{spec}}$
Dropper + Collector	1.0	1.0	1.0
Generalist + Generalist	0.0	0.0	0.0
Inefficient Dropper + Collector			
Dropper + Collector + Hitch-hiker			
Swapper + Swapper			
Lazy Generalist + Lazy Generalist			

Team Strategy	$\boldsymbol{R_{coop} \times P}$	$\boldsymbol{R'_{coop} \times P}$	$\boldsymbol{R_{spec} \times P}$
Dropper + Collector	1.0	1.0	1.0
Generalist + Generalist	0.0	0.0	0.0
Inefficient Dropper + Collector			
Dropper + Collector + Hitch-hiker			
Swapper + Swapper			
Lazy Generalist + Lazy Generalist			

2.2 Experiments

Now, with a landscape analysis method and a means of quantifying specialisation, this brings us to the first hypothesis and experiment.

Hypothesis 1.1: Specialist solutions outperform generalist ones.

Experiment 1.1- Specialisation Plots of RWG Analysis: Use the chosen specialisation metric to assess all the solutions found by RWG analysis and incorporate them into the plots.

The most specialised solutions are expected to be at the rightmost of the mean plot and above the generalist ones. If this is the case, then it supports the hypothesis. If there are generalist solutions that outperform specialist ones, it disproves the hypothesis.

To analyse the SlopeForaging problem, I used the same method as Oller et al [1], comparing the same 3 architectures: a feedforward neural network (FFNN) with no hidden layers, one with 1 hidden layer composed of 4 hidden units and one with 2 hidden layers also composed of 4 hidden units. As in [1] I ran experiments with and without a bias neuron and found performance to be better without bias. All networks use tanh activation. For each architecture, I sample 10,000 genomes, where each genome contains weights for two neural networks, one for each agent on the team. For each sampled genome, I run 20 episodes of the simulation. I use a normal distribution with mean 0 and unit variance. Other distributions can be used instead of a normal one but this is the one that proved most effective in my experiments and those of Oller et al. Degree of specialisation is measured on a scale from 0 to 1 with 0 scores being represented as red and 1 scores represented as green and scores in between falling on the spectrum between those colours.

2.2.1 Results- Landscape

Looking at Figure 2 in comparison with the Gym benchmarks in Figure 1, we can immediately see that the SlopeForaging task is more difficult than all the standard Gym benchmarks. The mean score is below zero for all architectures and the distribution histogram for the architecture with two hidden layers

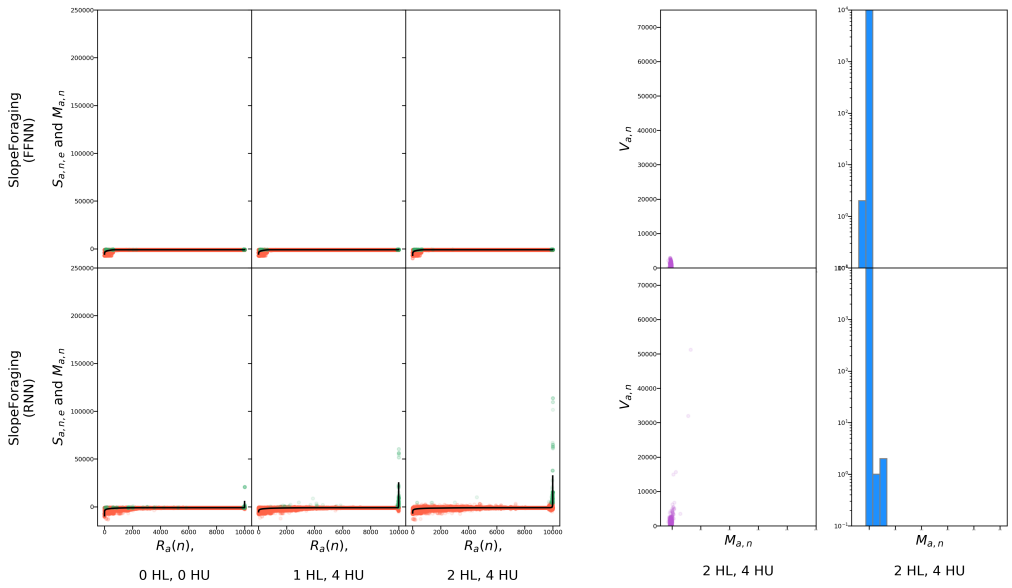

Figure 2: RWG Analysis without overlayed specialisation

shows that all episodes indeed receive negative scores. This holds whether or not there is a bias neuron as seen in Figure 12 in the appendix. The same architectures that can successfully solve the Gym benchmarks appear to be insufficient to solve the SlopeForaging problem, indicating that this problem is more difficult than all of Gym's classic control benchmarks.

When we observe the failed controllers to see where they go wrong, we see them carrying out repetitive unproductive behaviours or oscillating between states. For example, an agent at the nest might choose to do a pickup action, despite there being no resource in range. Since there is no resource in range, the state won't change and the agent's observation will be the same at the next time step and it will do the same action again, repeating itself until the final time step. Another agent might move forward when in the nest and move backward when in the cache, oscillating between two states until the end of the episode.

Using a FFNN means that if an agent makes a particular observation, they will perform the same action every time they make that observation, meaning it is easy for agents to get stuck. A recurrent neural network (RNN) performs

an action based on the observation as well as the last action, so if the observation is the same at time t as it was at $t-1$ then an agent can, in principle, do a different action than the one it did at $t-1$ as that one obviously did not change the state. Consider the previously mentioned agent that does a pickup action with no resources nearby and makes the same observation at the following time step. An FFNN-based agent will, by necessity, do that same action again. An RNN-based agent might do a different action because while the observation is the same, the action it did at $t-1$ may not be. Of course, there is a chance the network will have the same output for that observation regardless of what the action at $t-1$ was, but the chance of that happening is less than it is for an FFNN, for which it is a certainty.

I repeated the experiments with an RNN, with the same combinations of hidden layers and bias. The RNN achieves much higher scores than the FFNN. Similar to the FFNN experiments, most samples score poorly, but in contrast, we see that top ranked episodes receive non-negative scores, which are only possible if resources are successfully retrieved by the agents. The mean curve has a spike at the far right, which gets more pronounced as more hidden layers are added. We also see in the distribution histogram that there are samples whose mean scores fall into higher ranking buckets. Much like Oller et al [1] networks with no bias neuron outperform those with a bias neuron. Most interestingly, we see that some episodes score above 100,000. To get a sense of the scores a good team should be able to achieve, I hard-coded both generalist and specialist behaviours and averaged their scores over 30 episodes, varying the random seed. The top scoring sample gets very close to the performance of the best human-implemented solution. Thus, the SlopeForaging problem is simple enough that RWG with a relatively small neural network is enough to find one or two successful solutions. That said, SlopeForaging is more difficult than all its counterparts among the Gym classic control benchmarks, commonly used to test state of the art reinforcement learning algorithms [1]

Looking at the plots in Figure 1, the environment most similar to SlopeForaging is MountainCar [**?**]. The mean curve is flat with a sharp spike at the end and the distribution histogram has most samples fall in the leftmost bucket, with fewer and fewer falling into the higher ranking buckets. The reason most solutions score very poorly is because both the SlopeForaging and MountainCar environments have sparse rewards. In the case of Moun-

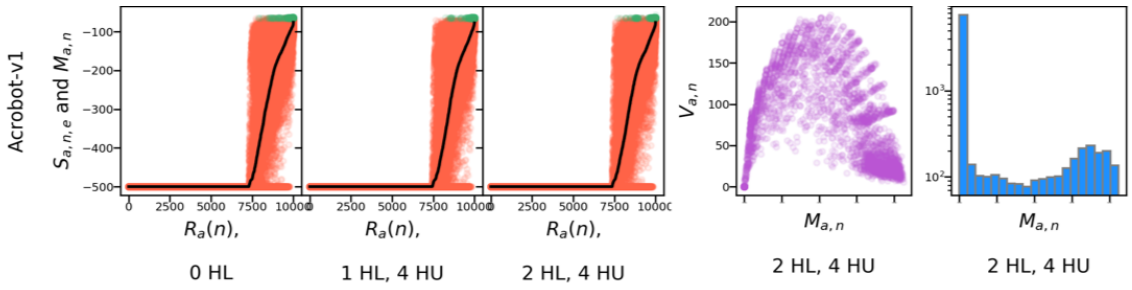

Figure 3: RWG analysis of Acrobot using an FFNN

tainCar, an agent must reach the top of the mountain to receive a score better than -200 and there are no rewards for partially successful intermediate behaviours. An agent that never moves is equivalent to one that stops just short of the goal. In SlopeForaging, an agent that travels up the slope, picks up a resource, carries it to the bottom of the slope and drops it just short of the nest is equivalent to an agent that oscillates back and forth between two tiles. A non-negative score can only be achieved if a resource is fully retrieved and there are no rewards for partially successful intermediate behaviours, so the core task of the environment cannot be learned incrementally.

Acrobot is another benchmark problem with a sparse reward. The goal of Acrobot is to swing the agent's arm up to a certain height within the allotted time. An agent that swings very close to that height is equivalent to one that does nothing. However, for Acrobot in Figure 3, there is a clearly visible gradient in the mean plot. This is because within the set of agents that complete the task, some complete it faster. The same applies to MountainCar and SlopeForaging. Within the set of agents that go up the mountain in MountainCar, some do it faster. Within the set of agent teams that can retrieve a resource, some retrieve more. So those mean plots also have a gradient, but a very faint one. Oller et al suggest that for environments with a large plateau and sharp spike in their mean curve, such as MountainCar, highly exploratory algorithms are likely the best choice, rather than gradient-based ones. In later sections, this research uses CMA-ES (as described in Section **??**) in part because it is gradient-free.

While SlopeForaging is similar to MountainCar and, to a lesser extent, Acrobot, the challenge posed by SlopeForaging is much more significant:

- *The task is multi-agent-* A minimum of two agents are needed to complete the task in the most effective way (i.e. cooperation) and because there is a non-cooperative solution to the problem, there are two stable equilibria that a learning algorithm can find.

- *The environment is stochastic* In a single agent environment, an agent's actions are the only thing that affect the state. In a multi-agent environment, the state is also dependent on the other agent(s), which means the environment is effectively stochastic. There can also be stochasticity in where new resources spawn, but only if the source has several empty spaces, which is not the case in our experiments.

- *The environment is partially observable-* Agents can only sense 1 tile in any direction (unless this parameter is changed). An agent can never truly know how its actions affect the state and oftentimes the state can change in ways that are relevant to an agent but the agent cannot detect it. Memory can help overcome this difficulty, which is likely why the RNN architecture is more successful.

- *The observation and action spaces are larger-* Looking at the observation and action spaces of all environments in Table 1 it's immediately apparent that SlopeForaging dwarfs all of its counterparts Even the simplest FFNN with no hidden layers or bias would require 14x6=84 weights per agent (168 for a team of two unique agents) compared to ((2+1)x4) + ((4+1)x4) + ((4+1)x3) = 47 for the most complex controller used by Oller et al to solve MountainCar or (4+1)x4) + ((4+1)x4) + ((4+1)x3) = 55 for Acrobot. The minimum weights required for each environment are enumerated in Table 2.

2.2.2 Results- Specialisation

In Figure 4 we see the same plots as before but with the R_{spec} measure of specialisation overlayed. The points on the plot are overwhelmingly red (i.e. generalist) with the majority of the green points being on the far right but a few being scattered farther left. This tells us that not all specialist solutions are high-performing, some achieve low scores. Observing one such team with low fitness but high specialisation, the video shows that all the resources they retrieve are retrieved in a specialised way. But they retrieve very few

Environment Name	Observation Space	Action Space	Observability	Reward Density
MountainCarContinuous	Box(2)	Box(1)	Full	Sparse
CartPole	Box(4)	Discrete(2)	Full	Dense
MountainCar	Box(2)	Discrete(3)	Full	Sparse
Pendulum	Box(3)	Box(1)	Full	Dense
Acrobot	Box(4)	Discrete(3)	Full	Sparse
SlopeForaging	MultiBinary(14)	Discrete(6)	Partial	Sparse

Table 1: Difficulty of each environment

Environment Name	Minimum Number of Weights
MountainCarContinuous	2
CartPole	8
MountainCar	6
Pendulum	3
Acrobot	12
SlopeForaging	168

Table 2: Number of weights required for the smallest possible network

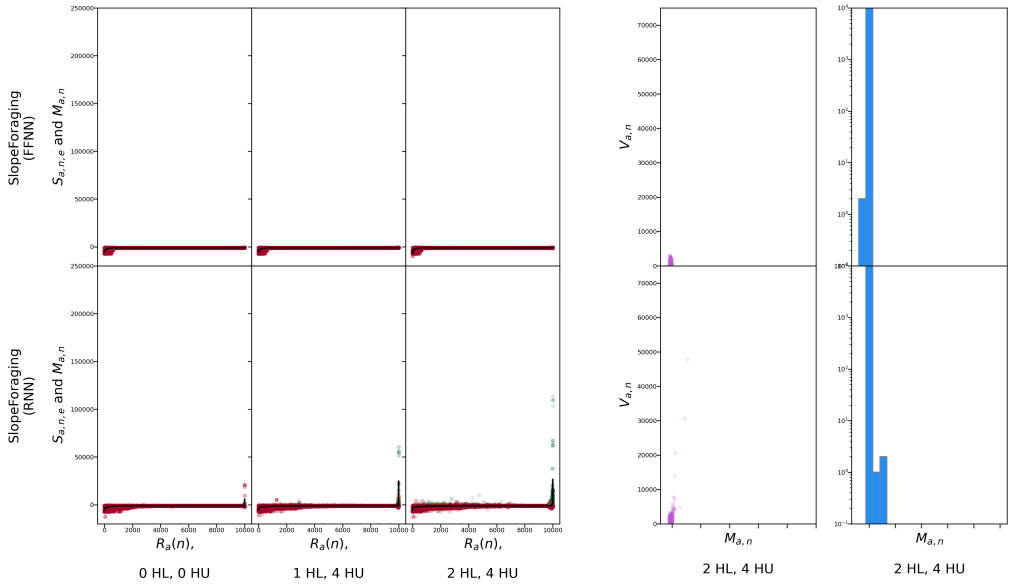

Figure 4: RWG Analysis with overlayed specialisation

resources and the dropper agent spends a lot of time moving up and down the slope ineffectively. This not only lowers the fitness, but in fact makes it even lower than a team that does nothing because going up the slope incurs a greater cost (-13) than moving aimlessly in the cache (-1) and moving down the slope (-0.2) is not enough to offset that cost. This is why there are some specialists among the worst performing genomes. However, what is also significant is that none of the top-scoring episodes are generalists. This suggests that while specialisation does not guarantee good performance, it may only be possible to achieve the highest scores using specialist strategies.

So Hypothesis 1.1 is partially supported:

Answer 1.1: Not all specialists are high-scoring, but the best solutions are always specialists.

2.3 Creating a Fitness Gradient

So far, the landscape of the slope foraging problem appears to be very flat, making it difficult to find any solution at all, let alone a specialised one. If a flat fitness landscape is inherent to task specialisation problems, then this is significant because it implies that algorithms that are highly exploratory should be used, as they are with similar problems [1]. However, it is also possible that the fitness function might be making evolution unnecessarily difficult because it only rewards task completion, treating partial success the same as failure. If partial successes are rewarded, the landscape could have a smoother gradient, making it easier for evolution to traverse. This leads to the following hypothesis:

Hypothesis 1.2: Rewarding agents for partial retrieval of the resource will create a smoother fitness gradient.

This hypothesis is answered by conducting the following experiment:

Experiment 1.2- RWG Analysis with Modified Fitness: Modify the fitness function such that agents are rewarded for partially retrieving a resource. Specifically, divide the reward for retrieving a resource by the length of the arena and give agents part of the reward if they move the resource a single tile closer to the goal. Then conduct RWG analysis and compare the plots to those produced for the original fitness function.

If the mean curve of the distribution has a smoother gradient, then the hypothesis is supported.

Looking at the 6 mean plots in Figure 5 compared to Figure 4, 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.

Also interesting is that the plot appears to be mostly red, meaning the solutions are predominantly generalists. This is not to say there are no special-

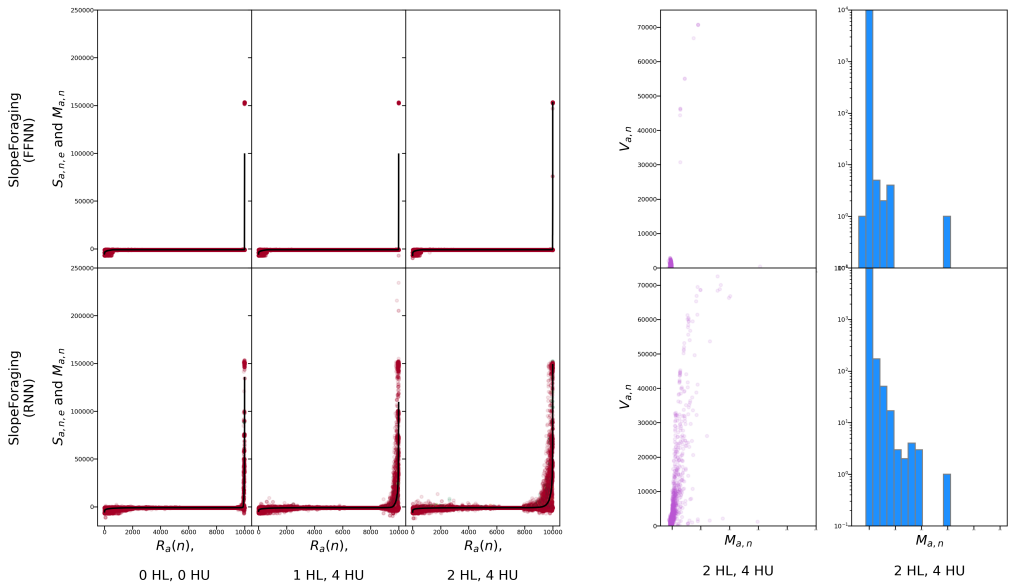

Figure 5: RWG Analysis with incremental rewards

ists. Rather, any team that brings resources back very close to the nest but doesn't quite deliver them would have received a negative score under the other reward structure, but now it has a positive reward, creating the gradient we see in Figure 5. However, all our specialisation metrics are dependent on resources being fully retrieved. If a dropper drops the resource, bringing it close to the nest, it gets points for the distance the resource travelled, but if the other agent doesn't deliver that resource to the nest, or picks it up but doesn't take it to the nest, the specialisation metric counts that as a generalist team. This is a shortcoming of the metric. Fixing this would mean either using a different type of metric entirely or adjusting the current metrics so that they count how many resources were picked up by multiple agents (regardless of whether or not they were delivered).

Instead, it is easier to select a team with high fitness and a specialisation score of 0 and visualise its behaviour. Doing this, we discover that the earlier prediction was in fact true. One agent behaves as a dropper, getting rewards for bringing the resource closer to the nest and reducing costs by not repeatedly going up the slope. The other agent stays still and does not retrieve any of the dropped resources. This is enough to get a very high

score, but is not counted as specialisation by the metric.
Thus the answer to Hypothesis 1.2 is:

Answer 1.2: 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 [11, 12]. 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" [11]. The bootstrapping problem often occurs when the goal behaviour is complex relative to the very simple available actions [12]. 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 [11]. 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.

2.4 Dimensionality Reduction

The neural networks used to solve the Slope Foraging problem have hundreds of weights, meaning that the fitness landscape has just as many dimensions and is consequently very difficult to visualise. Reducing the dimensionality of the problem to two dimensions (with a third dimension for fitness values) makes visualisation possible and gives many valuable insights into the features of the landscape, especially if the data-points are coloured according to the degree of specialisation. How far apart are specialist and generalist solutions in the landscape? How many optima are there? What trajectory does evolution take through the landscape?
The hypothesis is:

Hypothesis 1.3: There are two optima in the landscape, one composed of specialist solutions and one composed of mostly generalist solutions.

Experiment 1.3- Dimensionality Reduction: Assemble a dataset of solutions, combining some random ones from rwg and some evolved ones. Apply the chosen dimensionality reduction technique to the dataset. Use the chosen specialisation metric to plot the degree of specialisation (in addition to the fitness).

If there are indeed two optima, then the hypothesis is supported, otherwise it is not supported. If, additionally, the specialist solutions are fitter than the generalist ones, Hypothesis 1.2 is further supported.

2.4.1 Choosing a Dimensionality Reduction Algorithm

In [13], the authors use local optima networks [citation for LONs] to represent optima and t-SNE algorithm [14] for dimensionality reduction. But this technique only works for discrete problems. In [15] the authors adapt LONs to continuous landscapes as a proof of concept, though they mention that the resulting visualisation is highly dependant on the sampling method.
Sammon mapping [16] has been used for genetic algorithms [17] and neuroevolution [18, 19] in the past. It has often been chosen due to its ability to preserve relative distances between genomes in the reduced representation.
More recent work in the neuroevolution literature, though, [20, 21] uses t-

SNE [14].
The t-SNE algorithm was designed, in part, to overcome the "crowding" problem often faced by "local techniques for multidimensional scaling", where "the area of the two-dimensional map that is available to accommodate moderately distant datapoints will not be nearly large enough compared with the area available to accommodate nearby datapoint" so datapoints with small distances between them, if modelled accurately, means the moderately distant points are too far away and the spaces between clusters are too small, resulting in datapoints from different clusters all crowding together [14].
Though [20] visualises the space of behaviour characterisations (i.e. the space of phenotypes) rather than the fitness landscape (the space of genotypes). Additionally, their objective is to visualise the evolutionary trajectory rather than gain an understanding of the fitness landscape. Similarly, [21] use t-SNE to compress the observation space rather than the genotype space.
Recent work recommends using Latin Hypercube Sampling (LHS) in continuous spaces "to prevent over- sampling of the centre of the search space" [22]. LHS may also be better for high-dimensional spaces [23]
LHS divides the cumulative distribution function of each variable (in this case, each neural network weight) into bins equal to the number of samples [24]. It takes one sample from each bin and does the same for each weight. The samples for each weight are then combined randomly to form a genome.
Based on the literature, I have chosen to use t-SNE with latin hypercube sampling. For the purposes of this study, I only need to identify how specialist and generalist solutions relate to one another. Local Optima Networks, adapted to continuous landscapes, could be used for a more thorough analysis of the landscape, but is not strictly necessary and given that it is an emerging area of research, it is worthy of a research project unto itself and so is left for future work.
In LHS, sampled values must be within a certain range. I chose this range by taking 60 evolved team genomes and plotting the distribution of values for each genome, shown in Figure 6. According to the plot, all weights fall in the (-6,6) range. I tried sampling from this range as well as 5 others, decreasing it by 1 from each side each time. Ultimately, I obtained the best results using range (-5,5). The importance of choosing a range is discussed in [1] where the authors state that the "the bounds of the distribution (be there hard bounds as in the case of uniform distributions, or soft bounds as with the variance of a Gaussian) define the tested value range for the weights, which may be critical depending on application and network size".

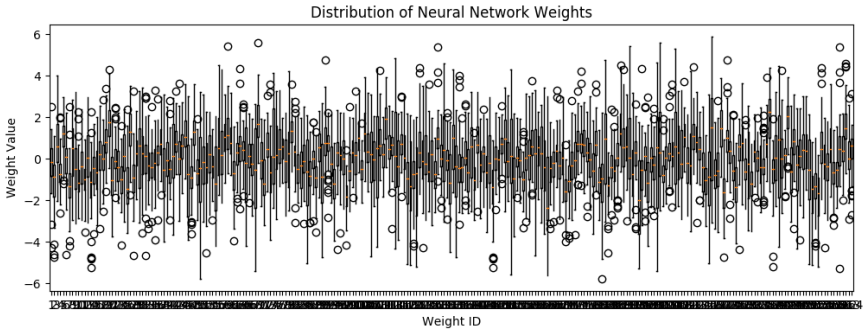

Figure 6: Distribution of weights in evolved genomes

2.4.2 Results

The algorithm I use to perform dimensionality reduction reduces the landscape to two dimensions. I then add a third dimension for visualising the fitness as height and I use colour to indicate the degree of specialisation. The plot in Figure 7 includes the best 60 and worst 60 genomes from lhs sampling with a range of (-5,5), along with 60 evolved genomes. There are only 120 rwg data-points so that there are not too many more random genomes than evolved genomes, which biases the dimensionality reduction and does not appropriately measure the landscape. High-scoring and low-scoring random genomes are chosen to represent a range of solution in the landscape. 30 of the evolved genomes were evolved using a centralised learner and re-evaluated to calculate their specialisation. The other 30 genomes were evolved using single-population co-evolution. Importantly, single-population co-evolution only outputs a single individual. If that individual is copied to form a team of two, the individual often does not perform as well with its clone as it did with the agents it was paired with during evolution. It is not clear why this is. Instead, I create a normal distribution where the mean is the individual genome and the variance is the sigma from the last generation of CMA-ES divided by 10 and sample two agents to create a team. Sigma needs to be divided by 10, otherwise solutions are low-quality, suggesting that particular peak in the landscape is very narrow. The plots in Figure 7 show the dimensionality reduction with just the rwg data, rwg and teams evolved using a centralised approach, and finally one with all data points. The plots also use three different specialisation metrics. t-SNE uses a perplexity value of 40 and 1000 iterations, with the rest of the parameters being the default from

the scikit-learn implementation of t-SNE.

The results show that there indeed appear to be two optima. One of the optima has a higher score while the other has a lower score . The high-scoring optima is mainly specialists and the low-scoring one is mainly generalists. The hypothesis is thus, supported.

Answer 1.3: There are two optima in the landscape, one composed of specialist solutions and one composed of generalist solutions.

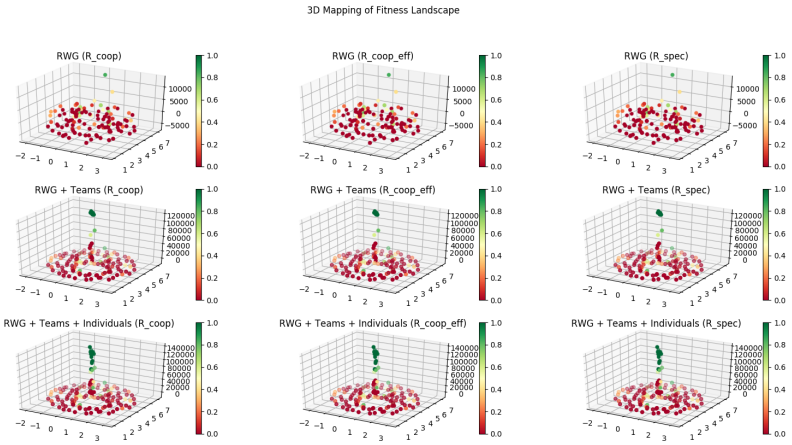

Figure 7: Reduced Fitness Landscape

2.5 Analysis of Different Slope Angles

For the Slope Foraging task, the slope is what provides evolutionary pressure for specialisation to emerge. Having no slope at all means there should be no benefit from specialisation and no pressure for it to evolve. Changing the slope angle changes how beneficial specialisation is. Experiments so far have only focused on one slope setting, so further experiments are needed to examine an environment with no slope and an environment with intermediate slope.
Thus:

Hypothesis 1.4.1: When the slope is flat, there will only be one optimum of generalists and no specialist solutions.

Hypothesis 1.4.2: An intermediate slope will have two optima (one specialist and one generalist).

Hypothesis 1.4.3: The specialist solutions will outscore the generalists, but the disparity will be smaller than the steeper slope setting.

Experiment 1.4- RWG with Different Slope Angles: Repeat RWG, as it was done for previous experiments, but with no slope and intermediate slope. Then perform 30 evolutionary runs, each with a different seed genome and random seed. Use the generated data to create RWG analysis plots and perform dimensionality reduction.

If there are any specialist solutions found for the 0 slope setting or more than one optimum, Hypothesis 1.4.1 is not supported, otherwise it is supported.

If the intermediate slope setting has more or less than two optima, Hypothesis 1.4.2 is not supported, if it has exactly two, it is supported. If the two optima for the intermediate setting are both generalist, both specialist or both a mix of the two, Hypothesis 1.4.2 is also not supported.

If Hypothesis 1.4.2 is supported and the specialist solutions do not outscore the generalist solutions, Hypothesis 1.4.3 is not supported. If the specialist solutions outscore the generalist solutions but by a margin that is greater than or approximately equal to the margin for the steep slope, Hypothesis 1.4.3 is not supported. If the specialist solutions outscore the generalists by a margin significantly less than the steep slope setting, the hypothesis is supported.

2.5.1 Results

We see in Figure 8 that for a sliding speed of 0 (i.e. flat slope) all solutions fall into the lowest bucket, with no episodes exceeding a score of 50,000. Additionally, there are no visible green data points, suggesting that none of the discovered solutions are specialists. The fittest genome is a generalist (where

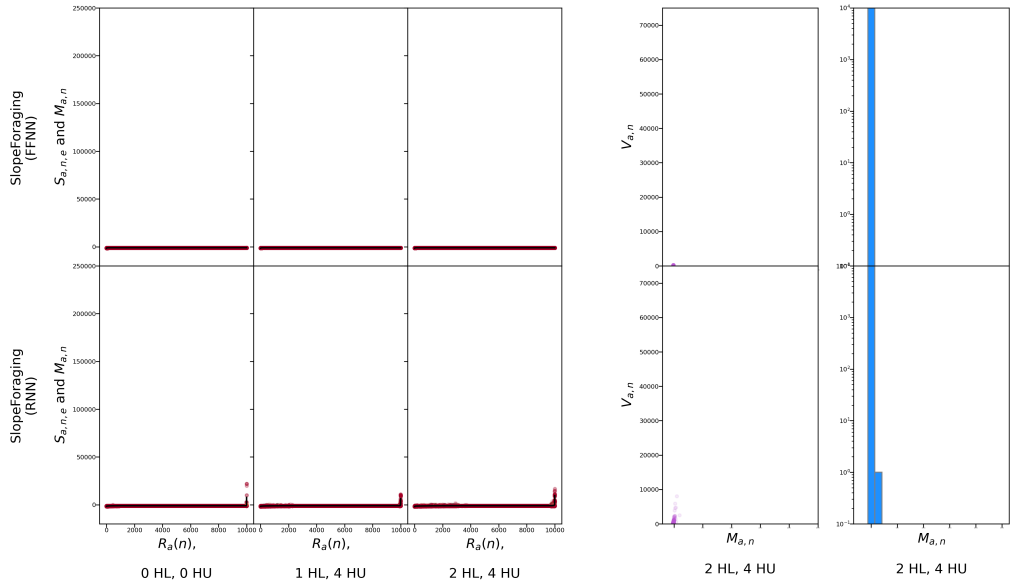

Figure 8: RWG Analysis for sliding speed 0

one of the agents does nothing) and the most specialised in the top 100 is a team where an agent picks up a resource another agent dropped, but because there is no slope, resources don't slide when dropped.

For additional clarity, I applied the same dimensionality reduction technique as in the previous section (including performing latin hypercube sampling and 60 evolutionary runs). The results in Figure 9 provide additional support for these findings. In the top row, there are no green data points in the two rightmost plots, but a single light green data point in the leftmost plot. The R_{coop} metric counts any cooperative transportation as specialisation whereas the R_{coop_eff} and R_{spec} metrics prioritise behaviours that drop on the slope and pick up on the cache, so there appears to be no specialisation.

Adding in the data-points from evolution, we see a single optimum emerge with the majority of solutions being generalists. When we observe the behaviour of the green data-points we see one agent dropping repeatedly on the slope then dropping on the cache. The other agent picks up from the cache, so it is counted as specialisation, even by R_{spec}.

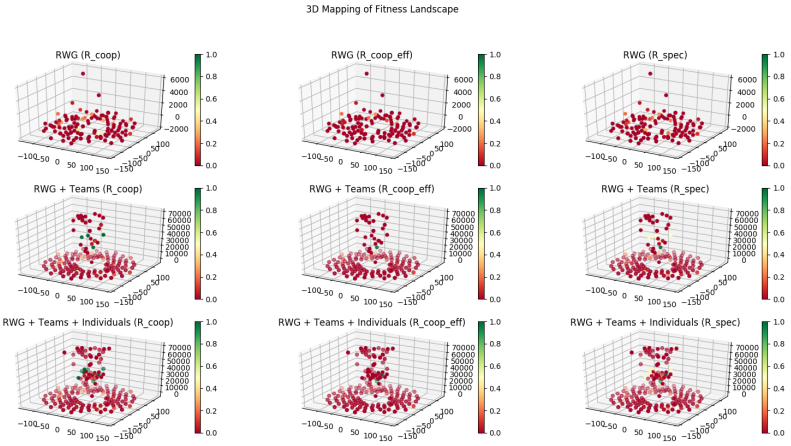

Figure 9: Reduced Fitness Landscape for sliding speed 0

In Figure 10 we can see there are more green points and they achieve a higher score. Observation of the behaviour confirms that they are specialists dropping and collecting as in the setup with sliding speed 4.
The dimensionality plot in Figure 11 is more unusual

2.6 Decentralised Solutions

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

Experiment 1.5- Decentralised Dimensionality Reduction: Perform 30 evolutionary runs of the decentralised setup. Calculate the degree of specialisation of all evolved solutions. Concatenate the genomes of all individuals on a team and apply dimensionality reduction to the data. Combine the decentralised data with the centralised data and plot on the same plot.

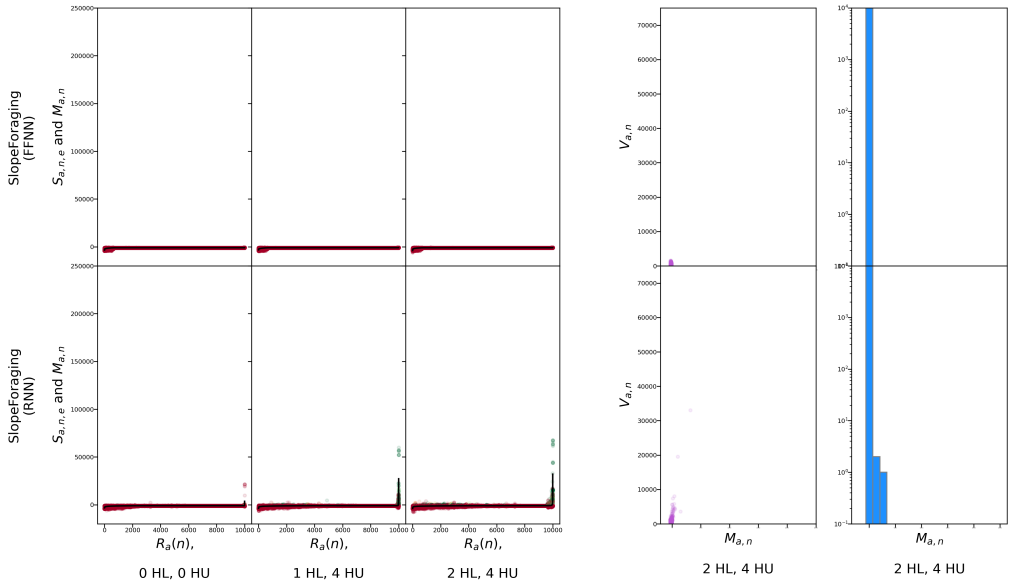

Figure 10: RWG Analysis for sliding speed 2

3 Centralised vs Decentralised Solutions

3.1 Background

In centralised learning, the entire team policy is learned by one learning process. In decentralised learning, each agent's policy is learned separately but concurrently (coevolution). In one-population coevolution, learning is centralised but it has some of the same game theoretic properties as decentralised learning.

When do centralised and decentralised solutions outperform one another in task specialisation problems?

3.2 Scalability

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 and it can also be expected that it will be more difficult to find specialisation,

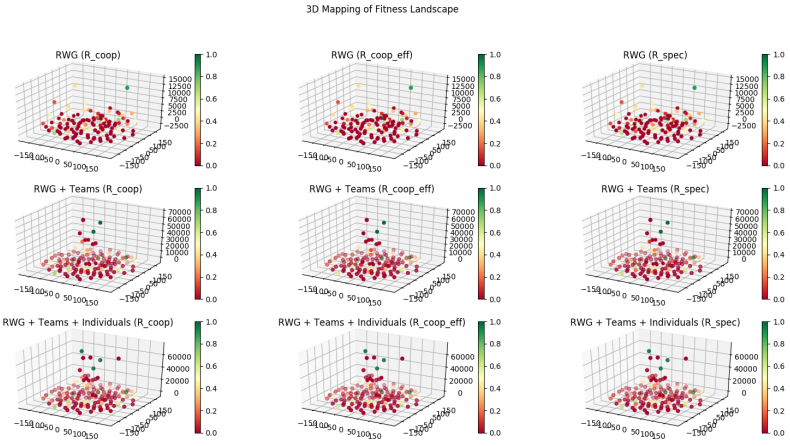

Figure 11: Reduced Fitness Landscape for sliding speed 2

if those solutions are indeed fitter. Since the decentralised approach is not expected to find specialisation in the first place, that is expected to continue for larger teams.

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. The time taken by the evolutionary process is not considered in the comparison. However, it is worth noting that the decentralised approach will likely have a slower wall-time if the evolutionary processes are not run in parallel, since evolution must be run once for each agent. Multi-threading should be implemented to avoid this problem otherwise computation time will scale linearly with the number of agents on the team.

Hypothesis 2.1.1: If evolution is run with larger team sizes, the fitness of centralised solutions will decrease.

Hypothesis 2.1.2: If evolution is run with larger team sizes, the fitness of decentralised solutions will decrease less rapidly than centralised solutions

Hypothesis 2.1.3 If evolution is run with larger team sizes, the centralised approach will find fewer specialists.

Hypothesis 2.1.4 If evolution is run with larger team sizes, the decentralised approach will find the same number of specialists.

Experiment 2.1- Evolution with Larger Teams: Do 30 evolutionary runs (centralised and decentralised) for teams of 4, 6, 8 and 10 agents. Calculate the specialisation of all evolved solutions. Plot the average fitness per agent and team specialisation for teams in each setup.

3.3 Robustness

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 team-mates. If an agent is evaluated with new team-mates 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 team-mates, the team performance will suffer.
Hypothesis 2.2.1: When evolved teams have team-members replaced, the average fitness per agent decreases for centralised teams.

Hypothesis 2.2.2: When evolved teams have team-members replaced, the average fitness per agent decreases for decentralised teams, but by a smaller margin than centralised teams.

Hypothesis 2.2.3: When evolved teams have team-members replaced, the team specialisation decreases for centralised teams.

Hypothesis 2.2.4: When evolved teams have team-members replaced, the team specialisation remains the same for decentralised teams.

Experiment 2.2- Changing Team-mates: Take all 60 evolved 2-agent teams (30 centralised + 30 decentralised). For each evolved team, replace

one agent 4 times: once with a dropper, once with a collector, once with a generalist and once with a stationary agent. Calculate the average fitness and team specialisation then repeat for the other agent on the team. That is a total of $2 \times 30 \times 4 \times 2 = 480$ fitness evaluations. Create a violin plot for the centralised setup and one for decentralised setup. Draw one violin for the original team and one for each replacement (a total of five). The data for each violin includes when each of the two agents was replaced. The y-axis shows the fitness and the plots also have a colour scale to show the degree of specialisation.

4 Theoretical Modelling

What is the price of anarchy in task specialisation problems?

Hypothesis 3.1.1: 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 3.1.2: Socially optimal teams are more likely to converge to the payoff-dominant Nash equilibrum while the self-interested teams are more likely to converge to the risk-dominant Nash equilibrium.

Experiment 3.1- Game-theoretic Modelling: Formulate a game that models the experimental setup. Find the nash equilibria and study the learning dynamics.

References

[1] D. Oller, T. Glasmachers, and G. Cuccu, "Analyzing reinforcement learning benchmarks with random weight guessing," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 975–982, 2020.

[2] G. S. Nitschke, M. C. Schut, and A. Eiben, "Evolving behavioral specialization in robot teams to solve a collective construction task," *Swarm and Evolutionary Computation*, vol. 2, pp. 25–38, 2012.

[3] J. Gautrais, G. Theraulaz, J.-L. Deneubourg, and C. Anderson, "Emergent polyethism as a consequence of increased colony size in insect societies," *Journal of theoretical biology*, vol. 215, no. 3, pp. 363–373, 2002.

[4] O. Gigliotta, "Equal but different: Task allocation in homogeneous communicating robots," *Neurocomputing*, vol. 272, pp. 3–9, 2018.

[5] G. Pini, A. Brutschy, M. Frison, A. Roli, M. Dorigo, and M. Birattari, "Task partitioning in swarms of robots: An adaptive method for strategy selection," *Swarm Intelligence*, vol. 5, no. 3-4, pp. 283–304, 2011.

[6] G. Pini, A. Brutschy, G. Francesca, M. Dorigo, and M. Birattari, "Multi-armed bandit formulation of the task partitioning problem in swarm robotics," in *International Conference on Swarm Intelligence*, pp. 109–120, Springer, 2012.

[7] E. Ferrante, A. E. Turgut, E. Duéñez-Guzmán, M. Dorigo, and T. Wenseleers, "Evolution of self-organized task specialization in robot swarms," *PLoS Comput Biol*, vol. 11, no. 8, p. e1004273, 2015.

[8] L. Li, A. Martinoli, and Y. S. Abu-Mostafa, "Learning and measuring specialization in collaborative swarm systems," *Adaptive Behavior*, vol. 12, no. 3-4, pp. 199–212, 2004.

[9] J. Gomes, P. Mariano, and A. L. Christensen, "Dynamic team heterogeneity in cooperative coevolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 6, pp. 934–948, 2017.

[10] S. O'Donnell and R. L. Jeanne, "Forager specialization and the control of nest repair in polybia occidentalis olivier (hymenoptera: Vespidae)," *Behavioral Ecology and Sociobiology*, vol. 27, no. 5, pp. 359–364, 1990.

[11] 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.

[12] 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.

[13] N. Veerapen and G. Ochoa, "Visualising the global structure of search landscapes: genetic improvement as a case study," *Genetic programming and evolvable machines*, vol. 19, no. 3, pp. 317–349, 2018.

[14] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[15] J. Adair, G. Ochoa, and K. M. Malan, "Local optima networks for continuous fitness landscapes," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1407–1414, 2019.

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

[17] 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.

[18] 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.

[19] F. Silva, P. Urbano, L. Correia, and A. L. Christensen, "odneat: An algorithm for decentralised online evolution of robotic controllers," *Evolutionary Computation*, vol. 23, no. 3, pp. 421–449, 2015.

[20] R. Wang, J. Clune, and K. O. Stanley, "Vine: an open source interactive data visualization tool for neuroevolution," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1562–1564, 2018.

[21] S. Risi and K. O. Stanley, "Deep neuroevolution of recurrent and discrete world models," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 456–462, 2019.

[22] G. Ochoa and K. Malan, "Recent advances in fitness landscape analysis," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1077–1094, 2019.

[23] M. A. Contreras-Cruz, G. Ochoa, and J. P. Ramirez-Paredes, "Synthetic vs. real-world continuous landscapes: A local optima networks view," in *International Conference on Bioinspired Methods and Their Applications*, pp. 3–16, Springer, 2020.

[24] J. C. Helton and F. J. Davis, "Latin hypercube sampling and the propagation of uncertainty in analyses of complex systems," *Reliability Engineering & System Safety*, vol. 81, no. 1, pp. 23–69, 2003.

A The Learning Process

Here "learning" refers to evolution.

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 Summary of Results

1 What are the fundamental features of the fitness landscape of a task specialisation problem?

Answer 1.1: Not all specialists are high-scoring, but the best solutions are always specialists.

Answer 1.2: Rewarding agents for partial retrieval of the resource creates a smoother fitness gradient, but the landscape is still largely flat.

Answer 1.3: There are two optima in the landscape, one composed

of specialist solutions and one composed of generalist solutions.

2 When do centralised and decentralised solutions outperform one another in task specialisation problems?

3 What is the price of anarchy in task specialisation problems?

C Additional Plots

C.1 Landscape Analysis

C.1.1 Experiment 1.1

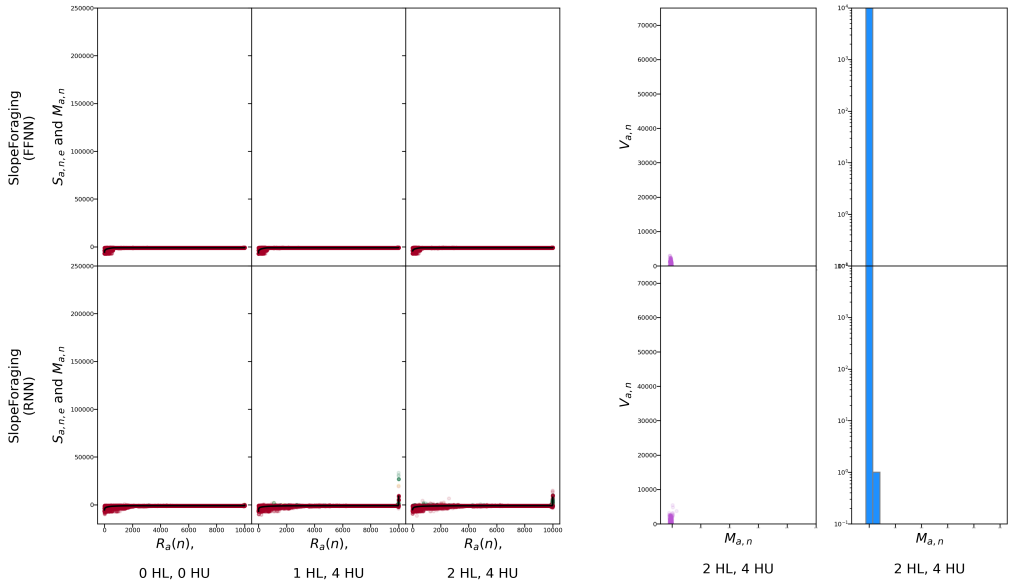

Figure 12: RWG Analysis with bias neuron for slope 4, using R_{spec} metric