Global structure of policy search spaces for reinforcement learning

B. Stapelberg
University of South Africa
Department of Decision Sciences
stapeb@unisa.ac.za

K.M. Malan
University of South Africa
Department of Decision Sciences
malankm@unisa.ac.za

ABSTRACT

Reinforcement learning is gaining prominence in the machine learning community. It dates back over three decades in areas such as cybernetics and psychology, but has more recently been applied widely in robotics, game playing and control systems. There are many approaches to reinforcement learning, most of which are based on the Markov decision process model. The goal of reinforcement learning is to learn the best strategy (referred to as a policy in reinforcement learning) of an agent interacting with its environment in order to reach a specified goal. Recently, evolutionary computation has been shown to be of benefit to reinforcement learning in some limited scenarios. Many studies have shown that the performance of evolutionary computation algorithms is influenced by the structure of the fitness landscapes of the problem being optimised. In this paper we investigate the global structure of the policy search spaces of simple reinforcement learning problems. The aim is to highlight structural characteristics that could influence the performance of evolutionary algorithms in a reinforcement learning context. Results indicate that the problems we investigated are characterised by enormous plateaus that form unimodal structures, resulting in a kind of needle-in-a-haystack global structure.

CCS CONCEPTS

•**Computing methodologies** → **Reinforcement learning;** *Discrete space search;*

KEYWORDS

reinforcement learning, fitness landscapes, local optima networks

ACM Reference format:
B. Stapelberg and K.M. Malan. 2019. Global structure of policy search spaces for reinforcement learning. In *Proceedings of the Genetic and Evolutionary Computation Conference 2019, Prague, Czech Republic, July 13–17, 2019 (GECCO '19),* 9 pages.
DOI: https://doi.org/10.1145/3319619.3326843

1 INTRODUCTION

Reinforcement learning (RL) is a training method based on rewarding desired behaviours and/or punishing undesired ones of an agent interacting with its environment. The agent learns by taking actions in the environment to maximize its cumulative long-term reward. RL is used in various fields such as operations research, information theory, game playing, control theory, simulation-based optimization, multi-agent systems and swarm intelligence.

In 2016 RL came into the spotlight when Google DeepMind's AlphaGo [24] program defeated the Go world champion, Lee Sedol. Due to promising results in areas such as controlling continuous systems in robotics [12], playing Atari [14], competitive video games [23, 29] and traffic light control [1], RL has experienced dramatic growth in attention and interest. The growth of the field through the number of RL-related publications per year is illustrated in [7]; almost 15000 in 2016 compared to less than 2500 in the year 2000.

The essential elements of an RL system are the agent in an environment, a set of actions that the agent can perform and a set of possible states of the environment. Within this, the agent has a policy, which is a mapping from every state that the agent can observe to a choice of action for the state. A policy can be thought of as a lookup table that associates an action with every possible state. A good policy will take an agent from the current state to a better state. The aim of RL is to find the policy that maximises the reward of the agent.

The most common algorithmic approaches to RL are methods that learn value functions. This means that the agent uses a particular model to estimate the long-term value of a sequence of actions from any particular starting state, allowing the expected reward of individual state-action pairs to be evaluated. In contrast, evolutionary methods do not learn value functions or evaluate individual state-action pairs, but instead consider the full search space of all possible policies. Each policy as a whole is evaluated only after a period of interaction with the environment.

There is a common belief that evolutionary methods are only effective if the space of policies is sufficiently small, or if the search space can be structured so that good policies are "easy to find" [28]. However, the question of what makes a problem easy to search for an evolutionary algorithm is not clear. The concept of fitness landscapes is used to understand this difficulty of finding good solutions in different problem search spaces [13]. Landscape features that have a strong influence on heuristic search are the number of local optima or peaks in the landscape, the distribution of the local optima in the search space, the correlation between fitness values of neighbouring points in the landscape and the topology of the basins of attraction of the local optima [17]. A further feature that can affect problem difficulty is high neutrality (solutions with the same fitness value) in the search space, which is common in difficult combinatorial search spaces. The fitness values are not changing in a neutral portion in a landscape, and this could therefore be mistaken for convergence on a local optimum.

GECCO '19, Prague, Czech Republic
 978-1-4503-6748-6/19/07…$15.00
DOI: https://doi.org/10.1145/3319619.3326843

This paper aims to investigate the nature of the full policy search spaces of RL problems as they would be experienced by evolutionary algorithms. This is a first step towards deciding when evolutionary approaches could be appropriate for solving RL problems. To achieve this, we investigate the global structure of the solution space of three simple RL problems using local optima networks [15], a technique that characterises the global structure of problems. A current disadvantage of this approach is that it requires a full enumeration of the search space. This study is therefore restricted to very simple problems, but still provides insight into the nature of policy search spaces.

Brief descriptions of RL, fitness landscapes and local optima networks are provided in Section 2. In Section 3 the three problem instances which are considered in this paper are discussed in the framework of RL and in Section 4 the fitness distributions for each problem instance are presented. Section 5 defines the neighbourhood for each problem instance and discusses the algorithmic approach to determining the local optima networks. Section 6 presents the findings of this paper, followed by a discussion and conclusion in Sections 7 and 8.

2 BACKGROUND

In this section we discuss key concepts of RL, introduce the concept of a fitness landscape together with a selection of features which will be used in this paper, and give an overview of local optima neutral networks and their components.

2.1 Reinforcement learning

Reinforcement learning is when an agent learns to complete a task by interacting with its environment and being rewarded for desirable behaviours or penalised for undesirable behaviours. The state in which the agent finds itself at each possible time step is therefore an important concept in RL. The observable or perceivable state of an agent can be thought of as the information that is available to the agent about its environment. If the agent can only observe part of its environment, the environment is referred to as partially observable. The world state is a complete description of the environment that the agent is in. If the agent can observe all aspects of its environment, the world state and observable state are the same, and the environment is referred to as fully observable. An RL agent must be able to sense the state of its environment to some extent and must be able to take actions that affect the state. The agent must have a goal or goals relating to the state of the environment.

An RL agent also has a policy, which can be thought of as a mapping from every possible state to the action that should be taken in that state. The policy determines the behaviour of the agent. The goal of RL is to learn the best policy to achieve a specific goal, which is referred to as the optimal policy. Policies that still result in goal achievement, but not as efficiently or effectively as the optimal policy, are referred to as sub-optimal policies.

2.2 Fitness landscapes

A discrete fitness landscape is formulated as a triplet (S, V, f) [20] where S is the set of all admissible solutions, $V : S \rightarrow \mathcal{P}(S)$ (where $\mathcal{P}(S)$ is the power set of S) defines a neighbourhood structure in the solution space, which is a function that assigns a set of neighbours $V(s)$ for every $s \in S$, and $f : S \rightarrow \mathbb{R}$ is a fitness function that assigns a solution quality to every $s \in S$. In this subsection we will discuss a selected number of fitness landscape features, but first we provide the definition of a local optimum, which is used in the discussion and in the rest of the paper.

Local optimum: A local optimum, which is taken to be a maximum here, is a solution s^* such that for every $s \in V(s^*)$, $f(s) \leq f(s^*)$.

Horn and Goldberg [8] define a local optimum as a point or region (a set of interconnected points with equal fitness) with fitness function value greater than those of all its nearest neighbours. This definition would consider flat plateaus at the tops of hills and ridges as single optima.

An RL problem is a type of optimisation problem since the learning agent has to maximise some reward function. A selected number of features of optimisation problems that could affect algorithm performance are discussed below. A detailed discussion on fitness landscape features can be found in the work of Malan and Engelbrecht [13].

Fitness distribution: The frequency with which each fitness value occurs in the search space S can be used to provide a profile of the problem. Due to the size of most search spaces, the fitness distribution of a problem cannot always be determined exactly and therefore sampling and grouping strategies are used to estimated the fitness distribution.

Modality: Functions which only have one local optimum are referred to as unimodal functions and the local optimum is then the global optimum. Multimodal functions have more than one local optimum.

Landscape structure of optima: Ochoa et al. [15] proposed a modelling technique for compressing the essential landscape features for combinatorial optimisation problems into a graph called a local optima network (LON). A LON characterises the structure of a landscape and the distribution of local optima, and is discussed further in Subsection 2.3.

Neutrality: A fitness landscape is neutral if there are a large number of solutions $s \in S$ which have neighbours with the same fitness value, i.e. if $x \in V(y)$, then $f(x) = f(y)$. The solutions x and y are then referred to as *neutral neighbours*. The landscape is then composed of several sub-graphs of configurations with the same fitness value. Neutrality manifests in features such as plateaus and ridges in a landscape.

Neutrality in a fitness landscape can have a profound effect on the success of search algorithms, since a neutral portion in a landscape could be mistaken for convergence on a local optimum. A fitness landscape with large masses of neutrality could then be difficult to search, due to the lack of information available to the algorithm.

2.3 Local optima networks

Local optima networks [15, 17] were inspired by the modelling of complex physical energy landscapes in chemical physics [27] and later those of small atomic clusters [6] by taking only the configurations that correspond to energy minima as vertices. When two energy minima are connected (i.e. when the energy barrier separating them is sufficiently low) an edge is traced between them. This idea is adapted for combinatorial search spaces, where local

optima are the vertices of the network and edges represent the possible search transitions between local optima.

To analyse the structure of a neutral combinatorial fitness landscape, the concept of a LON is extended to that of a local optima neutral network (LONN) in the work of Verel et. al. [16]. Definitions of the components of the LONN model are provided below. For the equivalent definitions for non-neutral landscapes, see [17].

Neutral network: A neutral network is a connected sub-graph of the search space whose vertices are configurations with the same fitness value. Two vertices in a neutral network are connected if they are neutral neighbours.

Local optimum neutral network (LONN): A neutral network is a local optimum if all the configurations of the neutral network are local optima. Note that this definition includes single solution local optima which are just local optima in the non-neutral case.

Basin of attraction of a LONN: The basin of attraction b_i of LONN_i is the set of all solutions $s \in S$ that could possibly reach LONN_i using a stochastic hill-climbing algorithm for a sufficiently large number of iterations.

Local optima network (LON): The local optima network $G = (N, E)$ is the graph where the nodes are the LONNs and there is an edge between nodes LONN_i and LONN_j if there are two solutions $s_i \in b_i$ and $s_j \in b_j$ such that $s_i \in V(s_j)$. A detailed definition of weighted edges can be found in [16].

3 PROBLEM INSTANCES

In this section the three problem instances which are considered in this paper are explained and discussed in the framework of RL.

3.1 Vacuum world

In their reference work "Artificial Intelligence: A Modern Approach", Russel and Norvig [21] use a very simple vacuum-cleaner world example to illustrate the ideas of intelligent agents. The world is illustrated in Figure 1.

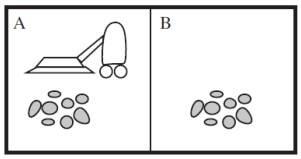


Figure 1: A vacuum-cleaner world with just two locations from [21, p. 36].

The environment has just two locations: squares A and B. The vacuum agent perceives which square it is in and whether there is dirt in the square. It can choose to move left, move right or suck up the dirt. The objective is to remove the dirt from both locations. Moving left in the left square, moving right in the right square and sucking in a clean square have no effect.

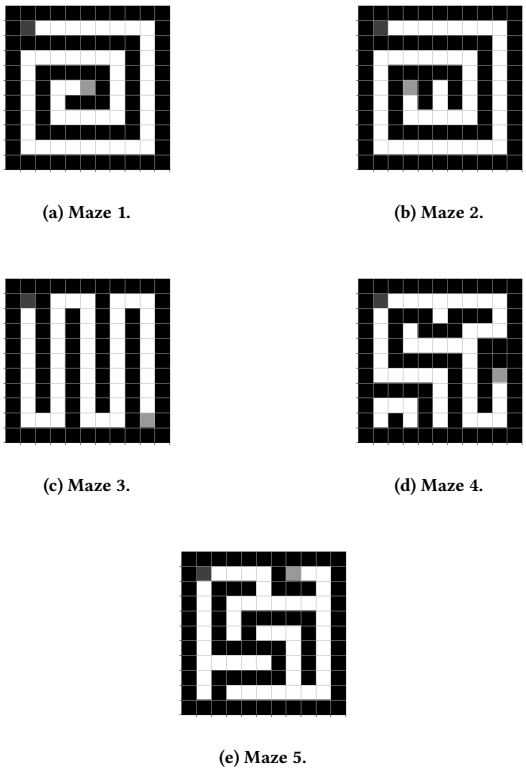
The agent can observe the whole environment, therefore the observable state and the world state are the same. The state is determined by both the agent location and the dirt locations and consequently there are $2 \times 2^2 = 8$ possible environment states. A policy in this example would be a list of eight actions, where each state has one action. The behaviour of the agent is deterministic with a policy always choosing the same action for a given state. Since there are three possible actions for each state: move left, move right and suck, the resulting policy space has a size of 3^8.

The initial state of the agent is illustrated in Figure 1: the agent is in location A and there is dirt in both locations.

The fitness of a policy is determined through simulated game play. The game terminates if the dirt in both locations has been sucked up or if the agent has executed ten actions. The fitness equals the number of actions before termination and is minimised. The optimal fitness has a value of 3 and is associated with the policy that results in the following sequence of actions from the initial state: suck, move right, suck.

3.2 Mazes

The second problem we consider is that of a simple maze puzzle. Due to their familiarity and intuitive nature, maze puzzles are often used for demonstrating and testing AI and RL techniques [21, 28]. The maze environment consists of a rectangular grid of cells, with an agent in a starting location and having a goal location. There are two types of cells: occupied cells (walls) and free cells (where the agent is allowed to move). In this paper we use five different maze puzzles of varying difficulty. The maze puzzles are provided in Figure 2, where the start location is illustrated by the dark grey cell and the goal location by the light grey cell.



(a) Maze 1. **(b) Maze 2.**

(c) Maze 3. **(d) Maze 4.**

(e) Maze 5.

Figure 2: The five instances of the maze puzzle used in this paper.

For this problem instance, the agent only has a partial view of the environment. It can perceive the cell in front of it, the cell directly on the left side of it and the cell directly on the right side of it. The actions available to the agent are taking a step forward, rotating left or rotating right.

The agent must navigate the maze to find the shortest possible path from the initial location to the goal location. The initial location is the same for all the mazes we consider: the top left corner of the maze.

The observable state of the agent is determined by the three cells that the agent perceives, and whether they are occupied (a wall) or not. For example, one observable state is where there is a wall in front and on the left of the agent, and the cell on the right is free. Therefore, there are $2^3 = 8$ possible observable states. A policy here would also be a list of eight actions, where each observable state has one action, and the behaviour of the agent is also deterministic. Since there are also three possible actions for each state, the size of the policy space is 3^8.

The fitness of a policy is determined through simulated game play. The agent is allowed 100 actions before termination. If the agent reaches the goal location, the game terminates. If the agent does not reach the goal location, the fitness is equal to the negative of the Manhattan distance (ignoring walls) between the agent's final location and the goal location. If the agent reaches the goal location, the fitness is equal to the maximum number of actions allowed minus the number of actions the agent took to reach the goal location. Here the fitness is maximised. Other measures for the fitness of unsuccessful policies can also be used, such as the number of cells along the shortest path (respecting the walls) from the agent's final location and the goal location.

The mazes that we consider are so-called perfect mazes [10]. A perfect maze is comprised of a start and end location and a spanning tree where every node is connected without cycles, and it is possible to reach one node from every other node. This means that there are no loops and unconnected areas in the maze, and the start and end location can be placed anywhere with a guaranteed unique solution.

3.3 Fruit collection task

In [11] a meta-algorithm for RL is introduced and evaluated on a fruit collection task. The next problem under consideration in this paper is this fruit collection task. The environment is a simple tabular domain: a 5×5 gridworld problem (see Figure 3), where the grey squares are occupied cells (walls) and the white circles are fruit. The goal is to collect the fruit placed at each corner as fast as possible.

The agent can observe the whole environment, therefore the world state is the same as the observable state. The agent has 18 possible positions to be in. There are four fruit to collect and the state of each fruit is either collected or not collected. The fruit can therefore be in 2^4 possible configurations. When all fruit are collected the game terminates, therefore there are $2^4 - 1 = 15$ non-terminal fruit configurations. This results in $18 \times 15 = 270$ possible states. The agent can choose to move North, East, South or West. A policy would be a list with length equal to the number of states,

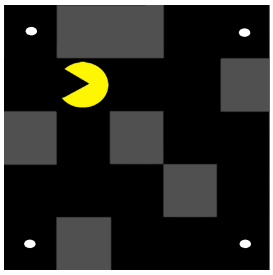
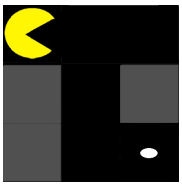


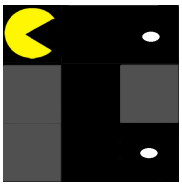
Figure 3: The Gridworld from [11, p. 9]

where each state has one option between any of the four possible actions. Therefore the size of the policy space is 4^{270}.

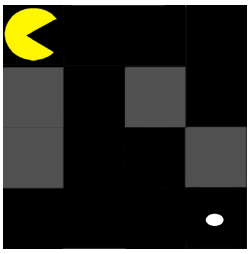
A full enumeration of a policy space of this size is computationally infeasible. The size of the policy space grows exponentially as the number of actions and/or states increase. We therefore consider simpler cases of the problem. The different cases are illustrated in Figure 4.



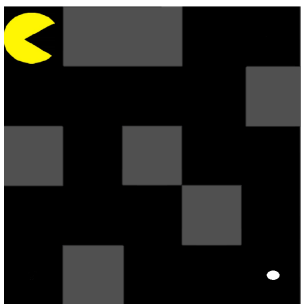
(a) Gridworld 1.



(b) Gridworld 2.



(c) Gridworld 3.



(d) Gridworld 4.

Figure 4: The four instances of the fruit task collection problem used in this paper.

The first two instances are 3×3 domains, with three occupied cells. The first instance (Gridworld 1) has one fruit in the bottom right corner and the second instance (Gridworld 2) has a second fruit in the top right corner of the grid. For the first instance there are six states and therefore 4^6 policies. For the second instance there are $(2^2 - 1) \times 6 = 18$ possible states and therefore 4^{18} policies.

The third instance (Gridworld 3) is a 4×4 grid, with four occupied cells and one fruit in the bottom right corner of the grid. There are 12 possible states and which results in a policy space with size 4^{12}.

The fourth instance (Gridworld 4) is a 5×5 grid with seven occupied cells as in Figure 3 and one fruit in the bottom right corner

of the grid. Here there are 18 possible states and consequently the size of the policy space is 4^{18}.

As before, the fitness of a policy is determined through simulated game play. The agent is allowed fifteen actions before termination. If all the fruit has been collected, the game terminates. The fitness equals the number of actions before termination and is minimised.

4 FITNESS DISTRIBUTION

A full enumeration of the policy search space of each problem instance is performed to construct a complete fitness landscape. In this section the fitness distribution of the various problem instances are presented and discussed. The fitness distribution for the problem instances is reported in Table 1. The optimal and sub-optimal policy percentages are provided. The policies with the worst fitness are the result of the agent not completing the task.

Table 1: The fitness distribution for a collection of problem instances

Instance	Size of policy space	Non-completing policy %	Sub-optimal policy %	Optimal policy %
Vacuum world	3^8	95.1%	1.2%	3.7%
Maze 1	3^8	95.884%	0.412%	3.7%
Maze 2	3^8	95.884%	0.412%	3.7%
Maze 3	3^8	99.588%	None	0.412%
Maze 4	3^8	99.588%	None	0.412%
Maze 5	3^8	99.771%	0.091%	0.137%
Gridworld 1	4^6	99.61%	None	0.39%
Gridworld 2	4^{18}	99.974%	0.002%	0.024%
Gridworld 3	4^{12}	99.951%	None	0.049%
Gridworld 4	4^{18}	99.997%	None	0.0003%

The fitness distribution for Maze 2 is provided in Figure 5, as an example to visually illustrate the proportions of the different policies. There are 3.7% of the policies that have the best fitness, 0.4% of the policies are sub-optimal and the remaining policies (95.884%) do not complete the task, with 69.1% of the policies ending at a Manhattan distance of 6 from the goal location and 26.7% ending at a Manhattan distance of 10 from the goal location.

From this data we observe that large proportions of the policy space will result in the agent not completing the task. Additionally, we observe that for all instances there are multiple optimal policies with the same fitness. The number of these global optima (optimal policies) could be large. For example, there are 4^{12} optimal policies for Gridworld 2.

It is also noted that even in the vacuum world problem instance where the environment is very simple, 95.1% of policies result in the agent not completing the task of cleaning all the dirt. The percentage of policies not completing the assigned task becomes even more with the increase in environmental difficulty. In the case of Gridworld 4 we see the worst ratio of 99.997%, which is particularly concerning if we consider the size of the policy space (4^{18}, approximately 68.7 billion), implying that when the policy space is searched, finding a policy that completes the task will be a difficult and time consuming process.

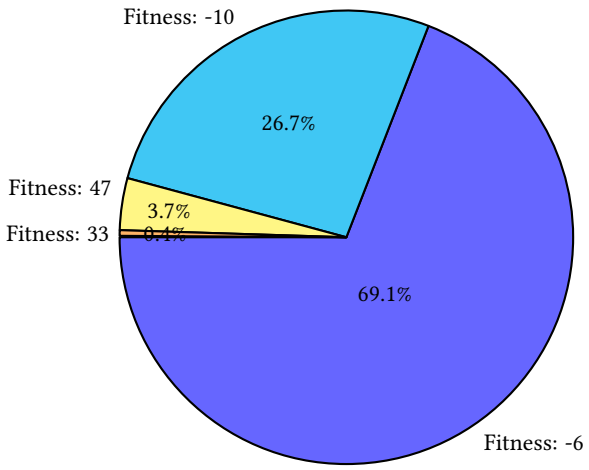


Figure 5: The fitness distribution for Maze 2

These observations lead to the investigation into how these optima are distributed in the policy space. Are the optima connected or far apart in the search space? These questions are investigated in the next section using the LONNs discussed in Subsection 2.3.

5 GLOBAL STRUCTURE ANALYSIS

In this section the concepts introduced in Section 2 are applied to the problem instances discussed in Section 3. We first define the the definition of a neighbourhood for each problem instance and then discuss how the LONNs are obtained.

5.1 Neighbourhood definition

The definition of a neighbourhood for each problem instance is now defined, which is used in determining the global structure of the policy space. In the present study, a policy is represented by a list of n_s actions, where n_s represents the number of states in the search space. Each action can take on n_a possible values. For example, in Gridworld 1, $n_s = 6$ and $n_a = 4$ (for North, South, East and West).

For the vacuum world and maze puzzle problem instances, the neighbourhood of a policy is the set of all policies that can be reached by changing a single component of the policy into a different action. For example, if the actions for the vacuum world instance are represented by the letters L (move left), R (move right) and S (suck), the policy *SRLRLSSS* will be in the neighbourhood of the policy *SLLRLSSS* but not in the neighbourhood of the policy *SRLLRRSS* (since the actions differ in three components).

For the fruit collection task problem, let the movement actions be represented by the letters N (North), E (East), S (South) and W (West). We define the neighbourhood of a policy as all policies that can be reached by changing a single component of the policy into a direction that is a single 90 degree turn away. For example, a movement N is one step away from the movements W and E, but S is two steps away. Mathematically this can be expressed as follows. Let the symbols (N, E, S, W) correspond to the integer

values $(0, 1, 2, 3)$, then the distance between two policies $P_1, P_2 \in S$ is defined as

$$d(P_1, P_2) = \sum_{i=1}^{n_s} d(P_{1,i}, P_{2,i}),$$

where $P_{j,i}$ is the ith component of the policy P_j, and

$$d(P_{1,i}, P_{2,i}) = \begin{cases} 1 & \text{if } \left|P_{1,i} - P_{2,i}\right| = 3, \\ \left|P_{1,i} - P_{2,i}\right| & \text{otherwise.} \end{cases}$$

The policy P_1 is a neighbour of policy P_2 if $d(P_1, P_2) = 1$.

5.2 Determining local optima neutral networks

In the case of non-neutral LON construction, the first step is to test each solution to find all local optima. When there is neutrality in the landscape, the neutral networks (that is, solutions of equal fitness that are connected by the definition of neighbourhood) introduce additional challenges because all of the solutions in a neutral network have to be considered in the process. The algorithmic approach to constructing a LON with neutrality is to first determine the neutral networks as sets of solutions. Once the neutral networks are determined, each neutral network is considered in order to determine whether it is a local optima plateau or not.

To find the neutral network that any policy belongs to, the neutral neighbours of the policy are determined. This is done by first generating all the neighbours of the policy and then determining which neighbours have the same fitness as the policy. The neutral neighbours are added to the neutral network set and the same procedure is followed with each element in the set. The process ends when there are no more neutral neighbours that are not already in the set.

If there is a policy in the neutral network that has a neighbour with better fitness, then the neutral network is not a LONN. If all the policies in the neutral network have better or equal fitness than any of their neighbours, then the neutral network is a LONN.

5.3 Local optima network construction

Once the local optima have been determined, the next step in LON construction is usually to portion the search space into basins of attraction for each optimum. Edges are then defined between local optima based on some notion of neighbourhood between basins. In this study, the landscapes were all found to be unimodal with neutral plateau global optima, resulting in a single basin of attraction. The construction of edges was therefore not required.

6 RESULTS

In this section the LONNs for each problem instance are determined and the results discussed. The neutral networks for each problem instance are provided in tables.

The vacuum world was found to have three neutral networks consisting of all policies. This means that no policy was not part of a neutral network. Table 2 presents the data on the neutral networks for the vacuum world problem. All optimal policies are in the same neutral network, all sub-optimal policies are in the same neutral network and all policies where the agent does not complete the task are in the same neutral network. There is only one LONN: the neutral network that consists of all the optimal policies. The LON is therefore a unimodal graph.

Table 2: The neutral networks for the policy space of the vacuum world

Neutral network	Size	Fitness
Optimal policies	243	3
Sub-optimal policies	81	4
Non-completing policies	6237	10

The neutral networks for the four fruit collection task problems are presented in Table 3. The structure of the problem is the same as for the vacuum world problem instance: the optimal policies result in a neutral network, the sub-optimal policies (if there are any) result in a neutral network and the policies where the agent does not complete the task result in a neutral network. Again, there is only one LONN which is the neutral network consisting of all the optimal policies, and the resulting LON is unimodal.

Table 3: The neutral networks for the policy space of the fruit collection task problem instances

Instance	Neutral network	Size	Fitness
Gridworld 1	Optimal policies	16	4
	Sub-optimal policies	None	None
	Non-completing policies	4080	15
Gridworld 2	Optimal policies	4^{12}	6
	Sub-optimal policies	4^{10}	8
	Non-completing policies	68.7×10^9	15
Gridworld 3	Optimal policies	8192	8
	Sub-optimal policies	None	None
	Non-completing policies	16769024	15
Gridworld 4	Optimal policies	2×4^{10}	8
	Sub-optimal policies	None	None
	Non-completing policies	68.7×10^9	20

In Table 4 the neutral networks for the collection of maze puzzles are provided. The same results as for the other problem instances are observed: there is only one LONN which is the neutral network consisting of all the optimal policies. The sub-optimal policies with the same fitness (if there are any) result in a neutral network for each fitness, but for this problem instance some of the non-completing policies with the same fitness result in more than one neutral network.

The reason for the mass connectedness of the policies with the same fitness, is that in these problems only certain elements of a policy have an effect on the optimality. For example, let us consider the fruit collection task problem, Gridworld 1. Each empty cell represents a possible position of the agent (see Figure 6 for the position numbers of each empty cell). Since there is only one fruit, the state space has size 6 (see Subsection 3.3) and therefore the length of a policy is 6. In position 1, the agent has to move east, since moving in any other direction will result in the agent staying in the same position. Once the agent is in position 2 it has to move south, since moving north will result in it staying in the same position, moving west will result in a loop between position 1 and

Table 4: The neutral networks for the policy spaces of a collection of maze puzzles

Instance	Neutral network	Size range	Fitness range
Maze 1	Optimal policies	243	44
	Sub-optimal policies	27	28
	Non-completing policies	6291	-8
Maze 2	Optimal policies	243	47
	Sub-optimal policies	27	33
	Non-completing policies (2 networks)	4536, 1755	-6, -10
Maze 3	Optimal policies	27	44
	Sub-optimal policies	None	None
	Non-completing policies (4 networks)	27, ..., 4536	-16, ..., -8
Maze 4	Optimal policies	27	72
	Sub-optimal policies	None	None
	Non-completing policies (8 networks)	27, ..., 3564	-13, ..., -3
Maze 5	Optimal policies	9	82
	Sub-optimal policies	6	42
	Non-completing policies (12 networks)	3, ..., 3627	-14, ..., -2

2 and moving east will result in a loop between position 2 and 3 (if the agent then moves west in position 3). Following this logic, if we represent the moves by the symbols N (North), E (East), S (South) and W (West), then an optimal policy will have to be of the form

State 1	State 2	State 3	State 4	State 5	State 6
E	S	□	S	E	□

where the □ can be any of the four possible actions. Since there are two components of the policy that can take on any of the four possible actions, there will be $4^2 = 16$ optimal policies. Due to the definition of neighbourhood, all of these policies are direct or indirect neighbours and therefore connected in the same neutral network.

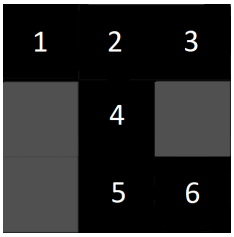


Figure 6: The position placement for Gridworld 1.

In a similar way, the non-completing policies share common actions in specific components of the policy, resulting in connectedness of policies with the same fitness value.

To imagine what these landscapes could look like, the data of Maze 2 (shown as a pie chart in Figure 5) is visualised as a two-dimensional continuous fitness landscape with neighbourhood in Figure 7. To visualise the size of the search space, the number of policies in Maze 2 (6561) was used as the domain of the two continuous variables. The proportions of solutions with each fitness value were mapped to equivalent ranges of these two variables. Approximately 26.7% of the search space has the worst fitness of -10 (shown in black in Figure 7) and is connected to a plateau with fitness -6, containing approximately 69.1% of the policies. This plateau is in turn connected to the sub-optimal plateau with fitness 33 containing 0.4% of the policies, which is visible in the change in colour from purple to orange in Figure 7. The optimal neutral network containing 3.7% of the policies is visualised as the small yellow plateau in Figure 7.

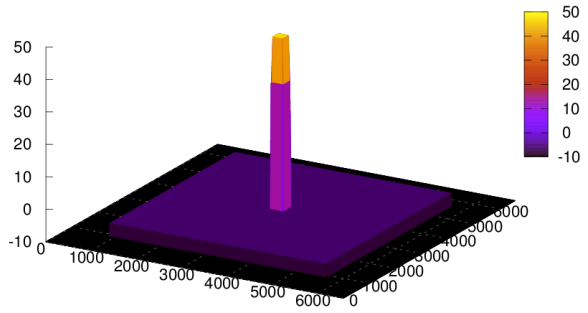


Figure 7: Two-dimensional plot of a continuous version of the search space for Maze 2

In summary, the LONs for all the problem instances are unimodal, with the LONN node consisting of all the optimal policies. The neutral networks for the non-completing policies can be large and the neutral networks for the optimal and sub-optimal policies are relatively small.

7 DISCUSSION

We have identified that the landscape structure of the RL problems under consideration are unimodal and contain large degrees of neutrality. It should be reiterated, that although there are many optimal policies for each problem instance, since they are connected in the same neutral network they are regarded as unimodal [8]. Given this information, the questions are then which search algorithms are best suited to solve RL problems with this type of global structure and when do search algorithms have to be adapted to effectively solve problems with certain features, for example high neutrality.

Recently, evolutionary computation has been shown to be of benefit to RL in some scenarios [5, 22, 30]. Studies have shown that the performance of evolutionary computation algorithms is influenced by the structure of the fitness landscapes of the problem being optimised. Horn and Goldberg [8] show that there are problems with maximum modality (such as their one-max function with "bumps") that are easy for a genetic algorithm to optimise and there are problems with minimal modality (such as long path problems) that are hard for a genetic algorithm to optimise. The global optima in the problem instances in this paper are isolated, resulting in a kind of needle-in-a-haystack problem, since the basin

of attraction is large. Rana [19] studied the effect of multimodality on genetic algorithm performance and found that although the number of local optima did not always affect genetic algorithm behaviour, highly fit local optima, particularly with large basins of attraction (which is present in the problems in this paper), did present a problem for genetic algorithm search.

Other than the number of global optima in the search space, neutrality in the search space can have a profound effect on the success of search algorithms [2, 18, 25]. If one chooses to use a evolutionary search approach to solve the problem instances investigated in this paper, the issue of neutrality has to be effectively dealt with. Specifically, evolutionary search techniques need to be modified to work effectively when problems have significant levels of neutrality [3]. For example, Owen and Harvey [18] show that a simple adaptation to the particle swarm optimisation algorithm - updating the global best position when a different solution of the same fitness is found - resulted in significant improvements in performance on problems with high levels of neutrality. The idea is to introduce the possibility of 'neutral drift', where the algorithm moves across plateaus rather than stagnating, which is what would normally happen in standard evolutionary approaches. When there is large-scale neutrality, then algorithms should not only be adapted to avoid stagnation, but also to increase the speed of traversal across neutral areas [18]. A possible approach to achieving this in genetic algorithms is proposed by Stewart [26], where neutral solutions that are further away from the centroid of the population have a higher probability of selection for reproduction than neutral solutions closer to the centroid, resulting in a faster traversal across plateaus. In a study of neutral landscapes Beaudoin *et al.* [4] found that neutrality had a smoothing effect on problem difficulty in that adding neutrality to a deceptive landscape made the problem easier, whereas adding neutrality to an easy landscape made it harder (as measured by the fitness distance correlation difficulty metric [9]). The landscapes in this study are easy, but the masses of neutrality present in the landscapes make them difficult to search.

In order to be able to effectively select an optimisation approach for a given problem, either automated or manually, knowledge of the problem structure is often very useful, as is evident from the discussion above. However, in the area of RL there has been little work trying to characterise the structure of the solution space of the RL problems. Such characterisation of RL problem structure is vital if the automation of RL algorithm selection is to be successful.

8 CONCLUSION

The purpose of this paper was to conduct an analysis on the global structure of the solution space of three simple RL problems in order to investigate the landscape features of RL problems. The three simple problems considered in this paper is the simple vacuum world used in [21], a collection of simple perfect maze puzzles and the fruit collection task problem from [11]. A full enumeration of the policy search space of each problem instance was performed to construct a complete fitness landscape.

In each problem instance we found masses of neutrality, therefore for each fitness there are multiple policies. In each case the resulting LON is a unimodal graph, with the only node the LONN with a single basin of attraction, resulting in a kind of needle-in-a-haystack problem. The neutral networks for the non-completing policies can be very large and searching the policy space can be difficult. The use of evolutionary computation techniques to search the policy space is influenced by the structure of the fitness landscape, and the adaptation of algorithms could become necessary.

This paper is a preliminary investigation into the global structure of RL problems. Since a full enumeration of the policy space of these types of problems can become computationally infeasible, future work can include the investigation into sampling techniques for larger problem instances. Investigating problems with continuous policy search spaces to determine whether these masses of neutrality still occur is a next step in the investigation of RL problems and their global structure.

REFERENCES

[1] I. Arel, C. Liu, T. Urbanik, and A. Kohls. 2010. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems* 4, 2 (2010), 128–135.

[2] L. Barnett. 1998. Ruggedness and neutrality - the NKp family of fitness landscapes. In *Proceedings of the sixth international conference on Artificial life (ALIFE)*. MIT Press, Cambridge, MA, USA, 18–27. http://dl.acm.org/citation.cfm?id=286139.286143

[3] L. Barnett. 2001. Netcrawling-optimal evolutionary search with neutral networks. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*. 30–37. https://doi.org/10.1109/CEC.2001.934367

[4] W. Beaudoin, S. Verel, P. Collard, and C. Escazut. 2006. Deceptiveness and neutrality: the ND family of fitness landscapes. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. ACM, New York, NY, USA, 507–514.

[5] P. Chrabaszcz, L. Loshchilov, and F. Hutter. 2018. Back to Basics: Benchmarking Canonical Evolution Strategies for Playing Atari. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*. 1419–1426.

[6] J. P. K. Doye. 2002. The network topology of a potential energy landscape: a static scale-free network. *Phys. Rev. Lett.* 88 (2002), 238701.

[7] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. 2018. Deep Reinforcement Learning that Matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. 3207 – 3214.

[8] J. Horn and D. E. Goldberg. 1995. Genetic Algorithm Difficulty and the Modality of Fitness Landscapes. In *Foundations of Genetic Algorithms 3*, L. Darrell Whitley and Michael D. Vose (Eds.). Morgan Kaufmann, San Francisco, CA, 243–269.

[9] T. Jones and S. Forrest. 1995. Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms. In *Proceedings of the Sixth International Conference on Genetic Algorithms*. Morgan Kaufmann, 184–192.

[10] P. H. Kim and R. Crawfis. 2015. The quest for the perfect perfect-maze. In *Proceedings of the Computer Games: AI, Animation, Mobile, Multimedia, Educational and Serious Games Conference*. 65–72. https://doi.org/10.1109/CGames.2015.7272964

[11] R. Laroche and R. Fraud. 2018. Reinforcement Learning Algorithm Selection. In *Proceedings of the Sixth International Conference on Learning Representations*.

[12] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. 2016. Continuous control with deep reinforcement learning. In *Proceedings of the Sixth International Conference on Learning Representations*.

[13] K. M. Malan and A. P. Engelbrecht. 2013. A survey of techniques for characterising fitness landscapes and some possible ways forward. *Information Sciences* 241 (2013), 148–163.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, A. Antonoglou, A. Wierstra, and M. Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. arXiv:1312.5602.

[15] G. Ochoa, M. Tomassini, S. Verel, and C. Darabos. 2008. A Study of NK Landscapesfi Basins and Local Optima Networks. In *Proceedings of Genetic and Evolutionary Computation Conference*.

[16] G. Ochoa, S. Verel, F. Daolio, and M. Tomassini. 2011. Local Optima Networks of NK Landscapes With Neutrality. *IEEE Transactions on Evolutionary Computation* 15, 6 (2011), 783–797.

[17] G. Ochoa, S. Verel, F. Daolio, and M. Tomassini. 2014. *Recent Advances in the Theory and Application of Fitness Landscapes*. Springer, Chapter Local Optima Networks: A new model of combinatorial fitness landscapes, 233–262.

[18] A. Owen and I. Harvey. 2007. Adapting Particle Swarm Optimisation for Fitness Landscapes with Neutrality. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*. 258 –265.

[19] S. Rana. 1999. *Examining the role of local optima and schema processing in genetic search*. Ph.D. Dissertation. Colorado State University, USA. Adviser: Whitley,

Darrell.
[20] C. M. Reidys and P. F. Stadler. 2002. Combinatorial landscapes. *SIAM review* 44, 1 (2002), 3–54.
[21] S. Russel and P. Norvig. 2009. *Artificial Intelligence: A Modern Approach* (Third ed.). Prentice Hall, Upper Saddle River, N.J.
[22] T. Salimans, J. Ho, X. Chen, , and I. Sutskever. 2017. Evolution strategies as a scalable alternative to reinforcement learning. (2017). arXiv:1703.03864.
[23] V. d. N. Silva and L. Chaimowicz. 2017. Moba: A new arena for game AI. arXiv:1705.10443. (2017).
[24] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529 (2016), 484fi??489. https://doi.org/10.1038/nature16961
[25] T. Smith, P. Husbands, P. Layzell, and M. O'Shea. 2002. Fitness Landscapes and Evolvability. *Evolutionary Computation* 10, 1 (2002), 1–34.
[26] T. Stewart. 2001. Extrema selection: accelerated evolution on neutral networks. In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*. 25–29. https://doi.org/10.1109/CEC.2001.934366
[27] F.H. Stillinger. 1995. A Topographic View of Supercooled Liquids and Glass Formation. *Science* 267 (1995), 1935–1939.
[28] R. S. Sutton and A. G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press.
[29] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Kttler, J. Agapiou, J. Schrittwieser, and et. al. 2017. *StarCraft II: A New Challenge for Reinforcement Learning*. Technical Report. Collaboration between DeepMind and Blizzard. arXiv:1708.04782.
[30] D. G. Wilson, S. Cussat-Blanc, H. Luga, and J. F. Miller. 2018. Evolving Simple Programs for Playing Atari Games. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '18)*. ACM, New York, NY, USA, 229–236.