Deep reinforcement learning for the control of bacterial populations in bioreactors

Neythen J Treloar[1], Alexander J.H. Fedorec[1], and Chris P Barnes*[1,2]

[1]Department of Cell and Developmental Biology, University College London, London, WC1E 6BT, UK
[2]UCL Genetics Institute, University College London, London, WC1E 6BT, UK

Abstract

Multi-species bacterial communities are widespread in natural ecosystems. Engineered synthetic communities have shown increased productivity over single strains and allow for the reduction of metabolic load by compartmentalising bioprocesses between multiple sub-populations. Despite these benefits, co-cultures are rarely used in practice because control over the constituent species of an assembled community has proven challenging. Here we demonstrate the efficacy of approaches from artificial intelligence – reinforcement learning – in the control of co-cultures within continuous bioreactors. We first develop a mathematical model of bacterial communities within a chemostat that incorporates generalised Lotka-Volterra interactions. We then show that reinforcement learning agents can learn to maintain multiple species of cells in a variety of chemostat systems, with different dynamical properties, subject to competition for nutrients and other competitive interactions. Reinforcement learning was also shown to have the ability to maintain populations within more selective bounds, which is important for optimising the productivity of reactions taking place in co-cultures. Additionally, our approach was shown to generalise to systems of three populations. These systems represent a level of complexity that has not yet been tackled by more traditional control theory approaches. As advances in synthetic biology increase the complexity of the cellular systems we can build, the control of complex co-cultures will become ever more important. Data-driven approaches such as reinforcement learning will enable greater optimisation of environments for synthetic biology.

keywords: reinforcement learning; generalized lotka-volterra; chemostat; bioreactor control; artificial intelligence

1 Introduction

The ability to engineer cells at the genetic level has enabled us to make use of biological organisms for many functions, including the production of biofuels [1, 2, 3], pharmaceuticals [4] and the processing of waste products [5]. Most of this work has been done with bacteria and yeast, due to the diversity of metabolism and their relative simplicity, although mammalian and plant cells are also widely used.

Co-cultures consisting of multiple distinct populations of cells have been shown to be more productive than monocultures at performing processes such as biofuel production [2, 3, 6] and alleviate the problem of metabolic burden that occurs when a large pathway is built within a single cell [7]. For these reasons co-cultures will be fundamental to the advancement of bioprocessing. However, growing a co-culture presents its own set of problems. The competitive exclusion principle states that if multiple populations are competing for a single resource, the species with an advantage over the others will drive the others to extinction. For example, if two populations of cells are growing on the same medium in a chemostat the one with the higher growth rate will always out compete the other. This has been proven to hold in a chemostat where multiple cell populations are competing for a single substrate; at

*Corresponding author: christopher.barnes@ucl.ac.uk

most one population can survive indefinitely [8]. However, it has been shown that factors including spatial heterogeneity, multiple resources, or other inter-population interactions such as commensalism, mutualism, or predation can alter the competition dynamics in the system and allow competitors to coexist in a chemostat [9]. For the coexistence of multiple cell populations in a chemostat at least one such factor must be built into the system. Additional challenges are that the interactions between different populations of bacteria can make long term behaviour in a co-culture difficult to predict [10]; the higher the number of distinct populations, the greater the challenge becomes to ensure system stability [11].

Previous methods of co-culture population control have been engineered into cells genetically. For example, using predator-prey systems [12] or mutualism [7, 13]. However, processes such as horizontal gene transfer make the long term genetic stability of a population hard to guarantee [10], meaning that genetic control methods can become less effective with time. Another potential problem is the increased metabolic load imposed on a cell due to the control genes, which can leave less resources for growth and the production of useful products [14]. These downsides can be avoided by exerting control over the environment and this is the dominant approach in industry. Established techniques are Proportional-Integral-Derivative (PID) controllers [15], Model-Predictive-Controllers (MPCs) [16, 17, 18] or the development of feedback laws [19, 20, 21, 22]. All three approaches are model-based, that is they rely on the construction and calibration of a mathematical model of the system. System parameters such as flow rate and nutrient concentration are controlled to optimise the performance of the system with respect to the model. Reliance on a model means that any modelling error will have a negative effect on the efficacy of the control. Additionally, feedback control laws are usually developed with the aim of increasing the region of stability of an existing steady state of the system. This means that there is a reliance on the existence of a steady state at the desired system variables, which is not required with reinforcement learning.

Reinforcement learning is a branch of machine learning concerned with optimising an agent's behaviour within an environment. The agent carries out certain actions and will receive a reward depending on the result of those actions on the environment. An agent is trained for a certain number of episodes; an episode encompasses a series of states, actions and rewards, which ends in a terminal state or after certain amount of time has elapsed. The agent shapes its behaviour in order to maximise its future reward, known as the return. Reinforcement learning has previously been used with a wide range of systems including complex 3D video games [23], the control of a periodically kicked rotor [24] and the optimisation of chemical reactions [25]. In this study, we used a type of reinforcement learning called Q-learning.

The control scenario we developed used independent control of bacterial populations exerted through auxotrophy and regulation of the corresponding nutrient. Each nutrient was controlled in a simple, on-off manner similar to a bang-bang controller. At each time point the agent decides, for each nutrient, whether or not to add a fixed amount of that nutrient to the environment. This was kept simple to make future implementation in a real chemostat easier. As in most implementations of reinforcement learning, time, the environment's state and the agent's actions are discretised. In our application a point in state space is defined by the population levels of each of the multiple species of bacteria in the chemostat. The size of the state space is finite and the range of population levels that it encompasses are chosen for each system so that the dynamics can be captured well. The actions taken by the agent change the levels of nutrients flowing into the chemostat and a reward is given if the population levels of each species are within a predefined range. A visual representation of a two dimensional case of this is shown in Figure 1, where the grid represents the discretisation of the population levels of the bacteria $\mathbf{N}_1$ and $\mathbf{N}_2$. An agent can distinguish between squares on the grid, but not continuous positions within each square. A trajectory over four state transitions is shown. The blue circles represent the true, non-discretised, state of the system at a certain time point and the darkened squares show the discretised state that the agent sees. The red arrows show the trajectory of the system state between timesteps. The agent does not have direct control over the magnitude and direction of the arrows; they are a function of the actions the agent can take, in this case the levels of

nutrients flowing into the bioreactor. The agent aims to maximize the total reward it receives. This corresponds to moving into and staying within an area in state space that fulfils the reward conditions, shown by the green square. Two agents were developed and their performance was compared, one using a lookup table to store state-action values (LT) and one using a neural network (NN).

The two agents were trained and tested on mathematical models of our chemostat systems. The results presented here show the application of LT and NN agents on a variety of systems. First the maintenance of two bacterial populations is achieved on a system containing one auxotrophic population with a high maximum growth rate and one non-auxotrophic population with a lower maximum growth rate. The agents are then applied to a number of systems comprised of two populations auxotrophic for different amino acids. Firstly, it is shown that both populations of auxotroph can be maintained despite heavy Lotka-Volterra competition. Secondly, the two populations are maintained in a system where no coexistent steady states exist. Then it is shown that finer control can be exerted on a system by manipulation of the reward conditions. Finally, both agents are extended to a higher dimensional system of three populations of bacteria, and it is shown that the NN agent performs much better than the LT agent on this higher dimensional system. Compared to existing control methods, reinforcement learning is more general, and this allows control of a range of chemostat systems with different properties.

2 Results

2.1 A mathematical model of interacting bacterial populations in a chemostat

The first chemostat model considered was that of two bacterial populations, one with a higher maximum growth rate but a reliance on a nutrient (tryptophan) for growth, the other with a lower maximum growth rate, but no auxotrophy. Without any environmental control, the faster growing strain would out-compete the slower growing strain if a sufficient amount of tryptophan was present in the environment, otherwise the slower growing strain would win. A sketch of the system is shown in Figure 2A.

The bioreactor was modelled as a chemostat with dilution rate q. Fresh medium was assumed to be continuously flowing into the chemostat, containing the carbon source used by all bacteria and the nutrient. The flow out of the bioreactor then contains unused nutrients and bacteria. The rate of change of concentration of the common carbon source C_0 is given by:

$$\frac{dC_0}{dt} = q(C_{0,in} - C_0) - \frac{1}{\boldsymbol{\gamma}_{0,1}}\boldsymbol{\mu}_1\mathbf{N}_1 - \frac{1}{\boldsymbol{\gamma}_{0,2}}\boldsymbol{\mu}_2\mathbf{N}_2 \tag{1}$$

where $\boldsymbol{\gamma}_{0,1}$ and $\boldsymbol{\gamma}_{0,2}$ are the yield coefficients for bacterial population 1 and 2 growing on the common carbon source. Population 1 is the non-auxotroph, population 2 is the tryptophan auxotroph. $\mathbf{N}_1$ and $\mathbf{N}_2$ are the population levels, $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ are their respective growth rates and $C_{0,in}$ is the concentration of the carbon source flowing into the bioreactor. The rate of change of the concentration of tryptophan C_T is given by:

$$\frac{dC_T}{dt} = q(C_{T,in} - C_{T,t}) - \frac{1}{\gamma_T}\boldsymbol{\mu}_2\mathbf{N}_2 \tag{2}$$

where γ_T is the yield coefficient for $\mathbf{N}_2$ utilising tryptophan and $C_{T,in}$ is the concentration of tryptophan flowing into the bioreactor. Here we have assumed that the non-auxotroph is not using any of the exogenous tryptophan.

The growth rates $\boldsymbol{\mu}_1$ and $\boldsymbol{\mu}_2$ are given by the Monod equations:

$$\boldsymbol{\mu}_1 = \boldsymbol{\mu}_{max,1}\frac{C_0}{\mathbf{K}_{s0,1} + C_0} \tag{3}$$

$$\boldsymbol{\mu}_2 = \boldsymbol{\mu}_{max,2} \frac{C_0}{\mathbf{K}_{s0,2} + C_0} \frac{C_T}{K_T + C_T} \tag{4}$$

where $\boldsymbol{\mu}_{max}$ is a vector of the maximum growth rate for each species, $\mathbf{K}_{s0}$ is a vector of the yield coefficients for the carbon source for each of the two strains and K_T is the saturation constant for tryptophan for $\mathbf{N}_2$.

The rate of change of population level for each species is given by:

$$\frac{d\mathbf{N}_i}{dt} = \mathbf{N}_i(\boldsymbol{\mu}_i - q) \tag{5}$$

Here we assume that the cell death rate is negligible compared to the dilution rate and can be ignored. An investigation into the steady state values of the system was carried out, the full analysis can be found in Section S1, where it is shown analytically and with simulation that at steady state $\boldsymbol{\mu}_1 = \boldsymbol{\mu}_2 = q$,

$$C_0 = \frac{q\mathbf{K}_{s0,1}}{\boldsymbol{\mu}_{max,1} - q} \tag{6}$$

and

$$C_T = \frac{qK_T}{\frac{\boldsymbol{\mu}_{max,2} C_0}{\mathbf{K}_{s0,2} + C_0} - q} \tag{7}$$

A second, more general model of an arbitrary number of auxotrophs in a chemostat, with Lotka-Volterra competition was also investigated. A sketch of the (two species) system is shown in Figure 2B. The rate of change of the concentration of the carbon source used by all species C_0 is given by:

$$\frac{dC_0}{dt} = q(C_{0,in} - C_0) - \sum_i \frac{1}{\gamma_{0,i}} \boldsymbol{\mu}_i \mathbf{N}_i \tag{8}$$

where $\boldsymbol{\gamma}_0$ is a vector of the bacterial yield coefficients with respect to the common carbon source for each species, $C_{0,in}$ is the concentration of the carbon source in the flow in to the bioreactor, $\boldsymbol{\mu}$ is the vector of the growth rates for each species, $\mathbf{N}$ is a vector of the concentrations of each bacterial species and q is the dilution rate.

The rate of change of each nutrient $\mathbf{C}_i$, where i denotes the species is given by:

$$\frac{d\mathbf{C}_i}{dt} = q(\mathbf{C}_{i,in} - \mathbf{C}_i) - \frac{1}{\boldsymbol{\gamma}_i} \boldsymbol{\mu}_i \mathbf{N}_i \tag{9}$$

where $\boldsymbol{\gamma}$ is a vector of bacterial yield for each auxotrophic species with respect to their nutrient, $\mathbf{C}$ is a vector of the concentration of each nutrient flowing into the reactor and is the quantity controlled by the algorithm. This assumes that all the auxotrophs are independent, i.e. each nutrient is only used by one population of bacteria.

The growth rates of each bacteria were calculated using the Monod equation:

$$\boldsymbol{\mu}_i = \boldsymbol{\mu}_{max,i} \frac{\mathbf{C}_i}{\mathbf{K}_{s,i} + \mathbf{C}_i} \frac{C_0}{\mathbf{K}_{s0,i} + C_0} \tag{10}$$

where $\boldsymbol{\mu}_{max}$ is a vector of the maximum growth rate for each species, $\mathbf{K}_s$ is a vector of the values of $\mathbf{C}$ for which each species reaches half its maximum growth rate and $\mathbf{K}_0$ is a vector of the values of C_0 for which each species reaches half its maximum growth rate.

The interactions between the different species of bacteria were modelled using the generalised Lokta Volterra equations:

$$\frac{d\mathbf{N}_i}{dt} = \mathbf{N}_i(\boldsymbol{\mu}_i - q + \sum_j a_{ij}\mathbf{N}_j) \tag{11}$$

where $\mathbf{N}$ is the population vector and a_{ij} are the entries of the interaction matrix $\mathbf{A}$, which models the competitive ($a_{ij} < 0$) or cooperative ($a_{ij} > 0$) effect each species has on all others. The combination of the chemostat and Lotka Volterra equations gives a model in which competition for a common resource, C_0, is modelled explicitly and other competitive effects can be modelled through the Lotka-Volterra interaction matrix $\mathbf{A}$. An investigation into the steady state value of this system was carried out and can be found in Section S2. For the parameter values in Table S2 it was found that of one million simulations, only 37 lead to steady states and that all of them were locally stable. Simulation also showed that the parameter values in Table S3 lead to a system where no coexistent steady states were found.

2.2 LT and NN agents can maintain both populations of bacteria in a single auxotroph system with stable coexistence

Both agents were trained on the single auxotroph system. The full list of the parameters can be found in Table S1. The level of the common carbon source was deliberately set to be relatively low to induce competition between the populations and couple the control of species 1 to species 2. The effective control of bacterial populations in the absence of saturating substrate is crucial for industrial applications where efficiency is important.

Ten training runs were carried out for each agent. The agents only have direct control of the growth rate of one of the populations, the other population is controlled purely through competition for the common carbon source. At each timestep the agents choose to set the concentration of tryptophan flowing into the bioreactor to 0.3gL^{-1} or 0. A reward of 1 was received if both populations of bacteria were greater than 10000×10^6cells L^{-1} and a reward of -1 otherwise. The reward boundary was chosen here to give a slight buffer between the point at which a negative reward is received and actual extinction. This proved to be an effective way to allow the agents to learn to maintain the populations.

The summary results are shown in Figure 3. With all ten of the final population curves and state-action plots in Figures S3, S4, S5 and S6.

In Figure 3A and 3B we see that both agents have performed well, with both of them reaching the optimum reward and survival time by the end of training. The NN has learned slightly faster and performed slightly better in terms of both reward received and survival time during the intermediate stages of training. The variance between the ten runs is similar for both agents.

During the ten runs the agents converge to one of a few final population curves. In Figure 3C and 3D an example of the most common solution found by each agent during the ten runs is plotted. We can see that the solutions found by both agents are very similar, with both populations oscillating in anti-phase with each other, with the slower growing population 1 (blue) being at a higher average level than auxotrophic population 2 (orange). The only difference between the two agents is at the very beginning of the episode, but beyond this the population curves are identical. The peak of the oscillations of both agents were analysed and it was found that they remained constant over time to within five decimal places. This shows that they are indefinite, meaning that the primary goal of the long term survival of both bacterial populations alive despite competition for nutrients has been achieved.

The state-action plots in Figure 3E and 3F are also very similar. The agents both explored almost exactly the same region of state space with both agents, in general, learning to add tryptophan when the tryptophan auxotroph is in a low population state. It seems that both agents are making decisions based on the auxotroph population levels, with no consistent changes in actions seen in the $\mathbf{N}_1$ direction, especially in the case of the NN. This implies that the agents have learned to focus on one particular feature of the system that is important for decision making.

In Figure S3 we can see that all of the solutions found by the LT are extremely similar, exhibiting the same long term oscillations but with slight differences at the beginning of the episode. This is reflected in the value plots in Figure S4 where there is minimal variation between each one, with all of them showing the same general trend by learning to add the auxotrophic nutrient when $\mathbf{N}_2$ is low

and the decisions made having seemingly no dependence on $\mathbf{N}_1$. However none of the state-action plots are exactly the same, showing that there is degeneracy in the mapping from state-action plot to system behaviour.

In Figure S5 we see that there is considerably more variation in the population curves the NN converged to. Five of them (B, D, F, G and J) exhibit the same behaviour as those found by the lookup table. The other five show oscillations of a similar frequency, but with different amplitudes and average values. Notably, and unlike all the other population curves, four of these solutions (A, C, E and I) have the average value of $\mathbf{N}_2$ slightly greater than the average of $\mathbf{N}_1$. Again, none of the state-action plots (Figure S6) are exactly the same, but they all show the same general trend by learning to add the auxotrophic nutrient when population 2 is low and the decisions made having seemingly no dependence on $\mathbf{N}_1$. These results show that reinforcement learning can be applied to a system of bacteria interacting through competition for nutrients and control a system that has previously been tackled using feedback control [22].

2.3 The NN agent learns faster than the LT agent in a double auxotroph system with stable coexistence

A co-culture of two auxotrophs, one requiring arginine and the other requiring tryptophan, was modelled in a chemostat environment using parameters in Table S2. The system was defined such that both the NN and LT agents gained a reward of $+1$ if the populations of both bacteria were above 2×10^6cells L^{-1} and a reward of -1 otherwise. This reward threshold was again chosen to allow a buffer between the point at which the agents start receiving a negative reward and actual extinction. The Lotka-Volterra competition in this system has lowered the number of total cells that can grow, in turn lowering the average number of cells in the chemostat drastically, meriting a lower reward threshold. The agents now have control of both auxotrophic species through their respectively required nutrients. This allows explicit control of each species without the need to couple their behaviour through nutrient competition. All learning parameters were identical for both the NN and LT agents. Due to the increased competition from the Lotka-Volterra interaction matrix the average time both species are maintained when an agent is choosing random actions is much shorter, meaning that more training episodes were required for the agents to successfully learn. As such, the number of episodes was increased to 10000.

The overall results are shown in Figure 4. From Figure 4A and 4B it is clear that the NN agent learns faster, with significantly better performance in terms of survival and reward received in the early and intermediate stages of training. By the end of training both agents have learned to keep the two species of bacteria alive for 3000 hours on all of their runs. The LT agent exceeds the NN mean reward received by the end of training, but is well within the error bars of the NN agent. This variation in the reward received by the NN compared to the LT seems to arise towards the end of the episode. During the beginning and intermediate stages of the episode there is not a noticeable difference in the variability of the two agents, but during the final episodes the standard deviation of the LT solutions goes to 0, as all of them reach the optimum reward. Whereas the standard deviation of the NN's solutions seems to increase during the final episodes, implying that the solutions diverge slightly toward the end of training. It is probable that further optimization of the NN parameters will increase the stability of the training process and decrease this variability.

Ten runs were carried out for each agent and on each of those runs a population curve was plotted after training (Figures S7 and S9). Both agents showed a tendency to converge to one of a number of distinct population curves and both had a certain population curve that arose more often than all the others. The most common forms of solution found by each agent are shown in Figure 4C and 4D. These are qualitatively similar, both having sustained anti-phase oscillations with very similar amplitudes. Both solutions have the average number of arginine auxotrophic population 1 (blue) lower than tryptophan auxotrophic population 2 (orange). This is expected as population 2 has a lower nutrient saturation constant, meaning that if the two nutrient levels are equal, population 2 will

grow faster than population 1. The main difference is that the oscillations found by the NN agent have about half the frequency of those found by the lookup table. The peaks of the oscillations of both agents were re-analysed and it was found that they remained constant over time to within two decimal places. This shows that they are indefinite, meaning that the primary goal of the long term survival of both bacterial populations alive despite competition for nutrients has been achieved.

The state-action plots both agents converged to are also similar (Figure 4E and 4F). The two agents learn to add both nutrients when the population of both are at intermediate levels, with a tendency to add the nutrient for population 1 if it is at a low level and population 2 is not. However they differ in that the NN agent has learned to add both nutrients in most of the visited state space, adding only the nutrient for population 2 at one area of state space, whereas the LT agent has multiple areas in the region of state space where populations 2 is low an where it will only add tryptophan. There are some regions where the state-action plots seem counter intuitive, for instance in the LT's state-action plot, Figure 4E, at grid point (0,2) we would expect the nutrient for population 1 (blue) to be added, squares (0,7), (4,0) and (7,0) seem unexpected for similar reasons. The NN agent's state-action plot is intuitively sensible apart from the line from (0,4) to (0,6) where we would expect the nutrient for population two (orange) to be added. These plots also show that states in which both species have high population are never visited, this implies that the number of cells is limited by a carrying capacity of the system due to the competition for the common carbon source C_0 and the negative terms in the Lotka-Volterra interaction matrix $\mathbf{A}$.

All final populations and state-action plots are provided in Figures S7, S8, S9 and S10. There are three distinct population curves to which the LT agent converged to during training (Figure S7). The first kind of solution found is seen in both Figures S7A and S7C, the second in Figure S7B and S7E and all remaining runs are a third kind of solution. All the population curves are oscillatory with the period of oscillation of both populations being the same and both populations oscillating in anti-phase to each other. The first and second types of solution are very similar, with the only difference being a small region at the beginning of the episode. Together these solutions make up four of the ten runs. The third solution is very similar, with an almost identical amplitude but a slightly increased frequency. All of the solutions found by the LT agent are optimal with regards to the amount of reward received. The NN agent (Figure S9) finds qualitatively very similar solutions to the LT agent, all of the ten runs resulted in anti-phase oscillations of the two bacterial populations. The NN agent found five distinct solutions, more than the three found by the LT agent, showing that the neural network output was more variable. Again there was a most common population curve, this time making up six of the ten runs Figures S9D, S9E, S9F, S9H, S9I and S9J that is very similar to the solutions found by the LT agent. All of the remaining four are different with two of them Figures S9A and S9B being qualitatively similar to the LT agent results. However, two of these, Figures S9C and S9G, are different from the others found by the NN agent and the LT agent. Both have very high frequencies and interestingly are the only two solutions found where the average level of population 1 is above that of more competitive population 2. A key difference here is that two of the solutions found by the NN agent do not fully satisfy the reward conditions at every time point, however they do successfully keep both species alive indefinitely. There are no population curves that are common between the two agents and it is interesting that the characteristics of the population curves found are effected by whether a NN or LT is used to store state-action values.

There is considerable variation in the state-action plots (Figures S8, and S10), even between population curves that look identical. The state-action plots were analysed and it was found that for both the LT agent and NN agent there are no state-actions that are conserved over all the most common solutions. This shows that there is a great amount of degeneracy, with many different state-action plots leading to the same system behaviour as expected. Importantly, these results show that reinforcement learning can be successfully applied to a realistic model of a bioreactor to keep a system of competing bacteria in a state of coexistence, with interspecies interactions and competition over a common nutrient.

2.4 Controlling a double auxotroph system with no stable coexistence

A harder challenge, and one of great interest, is to be able control systems with no coexistent steady states. These systems are incompatible with the feedback controller approach that has been applied to a system of bacteria in a chemostat previously [20, 19, 21, 22]. Such a system can be constructed using the parameters shown in Table S3. The Lotka-Volterra competition coefficients, a_{11} and a_{22}, were set to zero. This system was tested with one million simulations with uniformly randomly chosen $\mathbf{C}_{in}$, and initial starting points of $\mathbf{N}_1$ and $\mathbf{N}_2$. The initial populations were sampled uniformly from $[0, 20)$ and the concentrations sampled uniformly from $[0, 1)$. None of the one million simulations lead to coexistent steady states.

Both agents were trained on this system and the results are shown in Figure 5. A reward of +1 was given if both species are over 100×10^6cellsL^{-1} and a reward of -1 otherwise. As previously, the reward threshold was chosen based on the population levels the system was able to maintain to allow a buffer between the point at which the agent gained a negative reward and extinction.

In Figure 5A we can see that there is high variation in both agents' performance in terms of both survival time and reward. The LT agent failed to reach the maximum survival time in one run, where the trained agent only managed to keep the bacterial populations alive for 84 hours this is apparent from the population curve in Figure S11E. The NN agent failed to reach the maximum survival time in two runs, with the shortest time being 1176 hours.

The population curves found by each agent, Figure 5C and Figure 5D, are similarly irregular, but the LT agent has converged on oscillations with a smaller period and amplitude. The oscillations are no longer strictly anti-phase and are much more turbulent than before. This, and the increasing variation between runs as the episodes progress, appears to show the signs of a system exhibiting chaotic behaviour.

The state value plots of the two agents look have some differences. The LT agent (Figure 5D) has roughly divided the state space into four quadrants, adding both nutrients when both strains are at a low population, the nutrient for one strain if it is scarce and the other is not, and if both strains are at a high population neither nutrient is added. The NN agent however has learned to add both nutrients even when both species are at high population levels, this is reflected in the population curves as a higher amplitude of oscillation. For both agents we also see much more thorough exploration of the state space compared to Section 2.3, as the relative lack of competition compared to the system in Figure 4 allows both species to exist at high populations simultaneously. From these results it is clear that reinforcement learning can be successfully applied to a system with no coexistent steady states.

2.5 Specifying target behaviours

Fine control over bacterial levels could be useful for optimising future bioprocesses. For example, a reaction pathway that is spread between two populations could require separate tuning of the population growths, and hence product formation, to maximise the overall rate. This could be done by controlling the populations of the two strains of bacteria containing each part of the pathway. Here we train the agents to keep the two populations with specified bounds.

The two reinforcement learning agents were tested on the system in constructed using the parameters in Table S3, but with uniform Lotka-Volterra competition $a_{11} = a_{12} = a_{21} = a_{22} = -0.0001$. The agents were trained with the aim of keeping the two populations within a specified range. The range required was 100×10^6cells L^{-1} $< \mathbf{N}_1 < 400 \times 10^6$cells L^{-1} and 400×10^6cells L^{-1} $< \mathbf{N}_2 < 700 \times 10^6$cells L^{-1}. To do this a reward of $+1$ was given if both of these conditions were fulfilled, and a reward of -1 given if either one wasn't fulfilled. This has the effect of drastically shrinking the area of state space in which a positive reward is received (Figure 6E and 6F).

In Figure 6A we can see that the NN agent has outperformed the LT agent during the early and intermediate stages of training in terms of reward. The performance of the LT agent, in terms of reward, drops significantly during the intermediate stages of training, then increases rapidly near the

end of training to reach the optimum reward, whereas the NN agent is slightly below the optimum reward at the end of training. It is probable that during the intermediate stages of training the LT agent has not been exposed to enough data to have good value estimates, whereas the NN agent doesn't seem to have this problem. Within and directly adjacent to the area of state space where a positive reward is received the state-action mapping shows some consistency between agents. However, outside of this area both agents have learned to do a single action; the LT agent to add no nutrients and the NN agent to add the nutrients for population 1. Neither of these actions make sense across all of the state space and if the system moved out of the vicinity of the reward area it would likely result in extinction.

The population curves are, as before, qualitatively similar. The populations oscillate completely within the specified range for the LT agent, and almost completely within the range for the NN agent. The LT agent's population oscillations are sightly out of phase, but the NN agent's are in phase. The important result here is that both agents have successfully exerted control over the system such that both species stay completely, or almost completely, within their specified ranges. This shows that reinforcement learning can be used to exert finer control over a bacterial community.

2.6 The NN agent outperforms the LT agent on higher dimensional systems

Control over more complex communities composed of larger numbers of distinct microbial strains is a challenge that has, so far, proven too difficult for other approaches. Here we show that reinforcement learning can be effectively applied to a co-culture composed of three species.

The system is defined as above, using the parameters in Table S3, with an additional third population of bacteria with $\boldsymbol{\gamma_3} = \boldsymbol{\gamma_2}$, and $\mathbf{K}_{s,3} = \mathbf{K}_{s,2}$, these were chosen to be equal to the parameters of the second population as reliable estimates of K_s and γ for a third amino acid could not be found in the literature. Then μ_{max} was chosen to be $\boldsymbol{\mu}_{max,3} = 0.7$ to differentiate the third strain of bacteria from the second. The additional strain has the effect of increasing the size state-action space by a factor of 40, meaning the lookup table is now forty times bigger. However the complexity increase to the neural network is much smaller, with the first layer increasing in size by a factor of ten and the output layer a factor of two. All other layers remain the same. Ten repeated training runs were undertaken for each agent (Figure 7) and each run had 75000 episodes to allow for more training on the larger state space. From Figure 7 we can see that in terms of both reward and time survived the LT agent has only managed to marginally improve performance over the initial stages of training where actions are being selected randomly. The performance of the NN agent is much better, the average time survived by the end of training is just under 1500 hours. However, the increased complexity of the system has meant that the NN agent was unable to consistently reach the optimum survival time of 3000 hours by the end of training. The population curves in Figures S19 and S20 further show that the NN agent was able to learn to maintain long term oscillatory populations more frequently with four of the ten runs of the NN agent lasting over 500 hours compared to one run for the LT agent.

Figure 7C and 7D show examples of the typical population curves converged to by each agent. The LT agent has converged to an oscillatory solution, but is unable to maintain the oscillations beyond 109 hours at which time one of the strains of bacteria dies out. The NN agent on the other hand is able to maintain the for a much longer time, in fact for this run all three bacteria were kept alive for the optimum time of 3000 hours. This shows that the NN agent has the potential to learn how to enable long term coexistence, and implies that with a longer training process could learn to do this consistently.

3 Discussion

We have introduced the idea of using reinforcement learning for the control of bacterial populations within bioreactors. A mathematical model of a multi-population system of bacteria growing in a chemostat was developed, which modelled nutrient competition explicitly, with other interspecies

interactions being modelled through the Lotka-Volterra interaction matrix. We then applied a model-free reinforcement learning approach, Q-learning, to see if these systems could be controlled. Two reinforcement learning agents were developed. One used a lookup table to store state-action values, the other a neural network. It was found that both of the agents could control a variety of different model systems. The agent based on a neural network was even able to tackle the more complex case of three species.

Our approach could be implemented on real bioreactor systems by first training on a calibrated model of the system, where parameters are estimated using time-series data, as has been done previously [26, 27, 28, 29]. This model would then be used to pre-train the algorithm to produce a state-action mapping. This mapping could be used to control the system as is or, more likely, the trained agent could continue learning when applied to the real system with a small explore rate thereby refining its behaviour on the real chemostat dynamics. This would be desirable as although the generalised Lotka-Volterra dynamics have been shown to successfully model many systems [26, 27, 28, 30, 31, 29]. Another advantage to the reinforcement learning approach is its flexibility; if the parameters of the system change over time, for example from genetic drift, a reinforcement learning agent can learn to account for this and adjust its behaviour accordingly.

There are a number of future directions for this work. One of the assumptions made during training was that the initial populations were within the bounds that were required to satisfy the reward conditions. This is not unrealistic for practical use, however, training based on randomly chosen starting points both within and outside the reward conditions is likely to lead to a more robust system behaviour and would be a good avenue to explore. Other future directions would be to investigate the stability and robustness of the resulting combined system of the chemostat and the state-action decisions. The combined system is a limit cycle exhibiting oscillatory behaviour. The dynamics of this combined system are governed by a set of differential equations, from the chemostat and Lotka-Volterra equations, and an algebraic equation in the state-action mapping $a = \pi(s)$. This combination results in a differential-algebraic system of equations (DAEs). In a system composed of two bacterial strains the state-action mapping is a two dimensional step function with each discrete square in state space mapping to an action. Techniques exist for the analysis of DAE systems [32]; in principle one could analyse the joint chemostat-controller system of DAEs to understand the nature of the oscillatory dynamics and how robust the system is to perturbations. Another direction for future work could be to apply model-based reinforcement learning where a model of the environment is learned and can be used to generate simulated experience which, in addition to real experience, informs the learned policy.

A potential weakness of the reinforcement learning approach is the computational cost in training an agent. For the simulations performed in this study, the training time was on the order of a few hours, but could potentially be longer if more than three bacterial populations are used. This is related to another possible disadvantage; if it was desired to have the agent continue learning on the real system after training on a model, a significant amount of computing power may be required to update the agent's value estimates at each time interval. Another limitation is that our model bacterial system requires auxotrophy as a control mechanism and supplying amino acids could be costly. However, the reinforcement learning approach could also be applied to more general multi-species communities which naturally rely on different growth substrates, such as the yeast-lactic acid bacteria community [33].

In summary, we have demonstrated the potential for control of multi-species communities using deep reinforcement learning. As synthetic biology and industrial biotechnology moves to more complex systems for the production of everything from fine chemicals to biofuels, engineering of communities will become increasingly attractive. We believe that leveraging new developments in artificial intelligence seem highly suited to the control of these complex systems.

4 Methods

4.1 Q-learning algorithm

Q-learning was used to estimate the desirability of each action when in an arbitrary state. A value is learned for every state-action pair and stored as a value function, see Section S3.7 for more detail. The values of each state-action pair, $Q(s, a)$, were learned according to the Q-learning update rule:

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha(r_t + \gamma \max_a Q(s_{t+1}, a)) \tag{12}$$

The term $\max_a Q(s_{t+1}, a)$, where a is an action that can be taken by the agent, gives an estimate of the total future reward obtained by entering state s_{t+1}. This is weighted by γ, the discount factor which dictates how heavily the possible future rewards weigh in on decisions. α is the Q-learning step size. With the equation in this form it is easy to see that the updated value of $Q(s_t, a_t)$ is a weighted average, weighted according to α, of the current value and the reward given by the transition to state s_{t+1} and the estimate of the future rewards received as a result of being in s_{t+1}. The discount factor was set to 0.9 and the step size α decayed logarithmically as episodes progressed as a decaying step size is a condition to ensure the convergence of Q-learning [34].

The policy $a = \pi(s)$ chosen was the commonly used ϵ-greedy policy. A random action is chosen with probability ϵ and the action $a = \max_a Q(s_t, a)$ is chosen with probability $1 - \epsilon$. The explore rate ϵ decayed logarithmically with as episodes progressed. ϵ-greedy is a widely used policy that has been proven effective [23, 35], this combined with its trivial implementation made it a good choice.

The reward function was always conditioned on the populations of bacteria. The reward conditions specified an area in state space where a positive reward was received, outside of this area a negative or zero reward was received. In the simulations where the main objective was to avoid extinction a buffer between the specified area in state space and the populations reaching zero was found to be effective.

The state variables considered by the algorithm were the populations of each species of bacteria and the actions were the concentrations of each rate limiting nutrient in the flow into the reactor. Each variable was discretised, meaning that each state-action pair was represented as a point in a discrete $2n$ dimensional space, where n is the number of species of bacteria in the system (there is a dimension for the state and actions for each species of bacteria). The number of state-action pairs the algorithm learns values for is hence $2^n g^n$, where 2 is the number of concentrations the agents could pick from at each decision and g is the number of discrete values on the population axis. The agents can only distinguish between a finite number of bacterial population states, which are contained within a pre-defined range. The upper bounds of this range were chosen depending on the system, with the aim that the range approximately spanned all the states that were reached during training. If a population state was reached that exceeded the upper bound it was discretised into the highest state. The lower bound was always zero. For every system considered here $g = 10$ and the discretisation was uniform.

The LT agents had $2^n g^n$ entries, one for each state-action pair. The neural network used by the NN comprised of four hidden layers of 50 nodes. Each node in the hidden layers used the ReLU activation function. The input layer had g^n nodes and the linear output layer had 2^n nodes. The Adam optimiser [36] was used due to its ability to dynamically adapt the learning rate, which is favourable when doing reinforcement learning with a neural network [37].

It was found that restricting the agent to make decisions and receive rewards only every three time steps, instead of every time step, was required for good training performance. This is called frame skipping and is commonly used in reinforcement learning [23, 38]. This allowed an action that was taken to have enough time to effect the system such that there is clear cause and effect between the action and its result on the system. This is important as it allows the agent to correctly associate a reward it receives to the action that caused it. The fact that a reward could only be received every third time step is why the optimal reward for the experiments was 1000, even though the episodes lasted 3000 time steps. This shows that the the NN agent has the potential to learn long term control

of this more complex system and it is expected that with refined training the results could be further improved.

4.2 Implementation

Python version 3.6 was used throughout (available at http://www.python.org). The odeint function of the SciPy library [39] was used to numerically solve all differential equations. The neural networks in the NN agents were implemented in Google's TensorFlow [40]. The steady state Jacobian calculations were done using SymPy [41]. Numpy was used throughout [42]. The code and examples are available as a Python module on github: https://github.com/zcqsntr/CBcurl

References

[1] Lee R Lynd, Mark S Laser, David Bransby, Bruce E Dale, Brian Davison, Richard Hamilton, Michael Himmel, Martin Keller, James D McMillan, John Sheehan, et al. How biotech can transform biofuels. *Nature biotechnology*, 26(2):169, 2008.

[2] Hyun-Dong Shin, Shara McClendon, Trinh Vo, and Rachel R Chen. Escherichia coli binary culture engineered for direct fermentation of hemicellulose to a biofuel. *Applied and environmental microbiology*, 76(24):8150–8159, 2010.

[3] Garima Goyal, Shen-Long Tsai, Bhawna Madan, Nancy A DaSilva, and Wilfred Chen. Simultaneous cell growth and ethanol production from cellulose by an engineered yeast consortium displaying a functional mini-cellulosome. *Microbial cell factories*, 10(1):89, 2011.

[4] Chris J Paddon and Jay D Keasling. Semi-synthetic artemisinin: a model for the use of synthetic biology in pharmaceutical development. *Nature Reviews Microbiology*, 12(5):355, 2014.

[5] M Fujita, M Ike, and S Hashimoto. Feasibility of wastewater treatment using genetically engineered microorganisms. *Water Research*, 25(8):979–984, 1991.

[6] Mark A Eiteman, Sarah A Lee, and Elliot Altman. A co-fermentation strategy to consume sugar mixtures effectively. *Journal of Biological Engineering*, 2(1):3, 2008.

[7] Kang Zhou, Kangjian Qiao, Steven Edgar, and Gregory Stephanopoulos. Distributing a metabolic pathway among a microbial consortium enhances production of natural products. *Nature biotechnology*, 33(4):377, 2015.

[8] GJ Butler and GSK Wolkowicz. A mathematical model of the chemostat with a general class of functions describing nutrient uptake. *SIAM Journal on applied mathematics*, 45(1):138–151, 1985.

[9] AG Fredrickson and Gregory Stephanopoulos. Microbial competition. *Science*, 213(4511):972–979, 1981.

[10] Katie Brenner, Lingchong You, and Frances H Arnold. Engineering microbial consortia: a new frontier in synthetic biology. *Trends in biotechnology*, 26(9):483–489, 2008.

[11] Ahmad A Zeidan, Peter Rådström, and Ed WJ van Niel. Stable coexistence of two caldicellulosiruptor species in a de novo constructed hydrogen-producing co-culture. *Microbial cell factories*, 9(1):102, 2010.

[12] Frederick K Balagaddé, Hao Song, Jun Ozaki, Cynthia H Collins, Matthew Barnet, Frances H Arnold, Stephen R Quake, and Lingchong You. A synthetic escherichia coli predator–prey ecosystem. *Molecular systems biology*, 4(1):187, 2008.

[13] Wenying Shou, Sri Ram, and Jose MG Vilar. Synthetic cooperation in engineered yeast populations. *Proceedings of the National Academy of Sciences*, 104(6):1877–1882, 2007.

[14] Gang Wu, Qiang Yan, J Andrew Jones, Yinjie J Tang, Stephen S Fong, and Mattheos AG Koffas. Metabolic burden: cornerstones in synthetic biology and metabolic engineering applications. *Trends in biotechnology*, 34(8):652–664, 2016.

[15] Rimvydas Simutis and Andreas Lübbert. Bioreactor control improves bioprocess performance. *Biotechnology journal*, 10(8):1115–1130, 2015.

[16] J Prakash and K Srinivasan. Design of nonlinear pid controller and nonlinear model predictive controller for a continuous stirred tank reactor. *ISA transactions*, 48(3):273–282, 2009.

[17] Guang-Yan Zhu, Abdelqader Zamamiri, Michael A Henson, and Martin A Hjortsø. Model predictive control of continuous yeast bioreactors using cell population balance models. *Chemical Engineering Science*, 55(24):6155–6167, 2000.

[18] S Ramaswamy, TJ Cutright, and HK Qammar. Control of a continuous bioreactor using model predictive control. *Process Biochemistry*, 40(8):2763–2770, 2005.

[19] Iasson Karafyllis, Georgios Savvoglidis, Lemonia Syrou, Katerina Stamatelatou, Costas Kravaris, and Gerasimos Lyberatos. Global stabilization of continuous bioreactors. In *American Institute of Chemical Engineers-Annual Meeting, Sn. Francisco, USA*, 2006.

[20] Hernán De Battista, Martín Jamilis, Fabricio Garelli, and Jesús Picó. Global stabilisation of continuous bioreactors: Tools for analysis and design of feeding laws. *Automatica*, 89:340–348, 2018.

[21] Frédéric Mazenc, Jérôme Harmand, and Michael Malisoff. Stabilization in a chemostat with sampled and delayed measurements and uncertain growth functions. *Automatica*, 78:241–249, 2017.

[22] Karlene A Hoo and Jeffrey C Kantor. Global linearization and control of a mixed-culture bioreactor with competition and external inhibition. *Mathematical Biosciences*, 82(1):43–62, 1986.

[23] Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *AAAI*, pages 2140–2146, 2017.

[24] Sabino Gadaleta and Gerhard Dangelmayr. Learning to control a complex multistable system. *Physical Review E*, 63(3):036217, 2001.

[25] Zhenpeng Zhou, Xiaocheng Li, and Richard N Zare. Optimizing chemical reactions with deep reinforcement learning. *ACS central science*, 2017.

[26] Richard R Stein, Vanni Bucci, Nora C Toussaint, Charlie G Buffie, Gunnar Rätsch, Eric G Pamer, Chris Sander, and João B Xavier. Ecological modeling from time-series inference: insight into dynamics and stability of intestinal microbiota. *PLoS computational biology*, 9(12):e1003388, 2013.

[27] Simeone Marino, Nielson T Baxter, Gary B Huffnagle, Joseph F Petrosino, and Patrick D Schloss. Mathematical modeling of primary succession of murine intestinal microbiota. *Proceedings of the National Academy of Sciences*, 111(1):439–444, 2014.

[28] Charlie G Buffie, Vanni Bucci, Richard R Stein, Peter T McKenney, Lilan Ling, Asia Gobourne, Daniel No, Hui Liu, Melissa Kinnebrew, Agnes Viale, et al. Precision microbiome reconstitution restores bile acid mediated resistance to clostridium difficile. *Nature*, 517(7533):205, 2015.

[29] Ophelia S Venturelli, Alex V Carr, Garth Fisher, Ryan H Hsu, Rebecca Lau, Benjamin P Bowen, Susan Hromada, Trent Northen, and Adam P Arkin. Deciphering microbial interactions in synthetic human gut microbiome communities. *Molecular systems biology*, 14(6):e8157, 2018.

[30] Jonathan Friedman, Logan M Higgins, and Jeff Gore. Community structure follows simple assembly rules in microbial microcosms. *Nature ecology & evolution*, 1(5):0109, 2017.

[31] Stefan Vet, Sophie de Buyl, Karoline Faust, Jan Danckaert, Didier Gonze, and Lendert Gelens. Bistability in a system of two species interacting through mutualism as well as competition: Chemostat vs. lotka-volterra equations. *PloS one*, 13(6):e0197462, 2018.

[32] C Gear. Simultaneous numerical solution of differential-algebraic equations. *IEEE transactions on circuit theory*, 18(1):89–95, 1971.

[33] Olga Ponomarova, Natalia Gabrielli, Daniel C Sévin, Michael Mülleder, Katharina Zirngibl, Katsiaryna Bulyha, Sergej Andrejev, Eleni Kafkia, Athanasios Typas, Uwe Sauer, et al. Yeast creates a niche for symbiotic lactic acid bacteria through nitrogen overflow. *Cell systems*, 5(4):345–357, 2017.

[34] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[35] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[36] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[37] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR, abs/1507.06527*, 7(1), 2015.

[38] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

[39] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001.

[40] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[41] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017.

[42] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

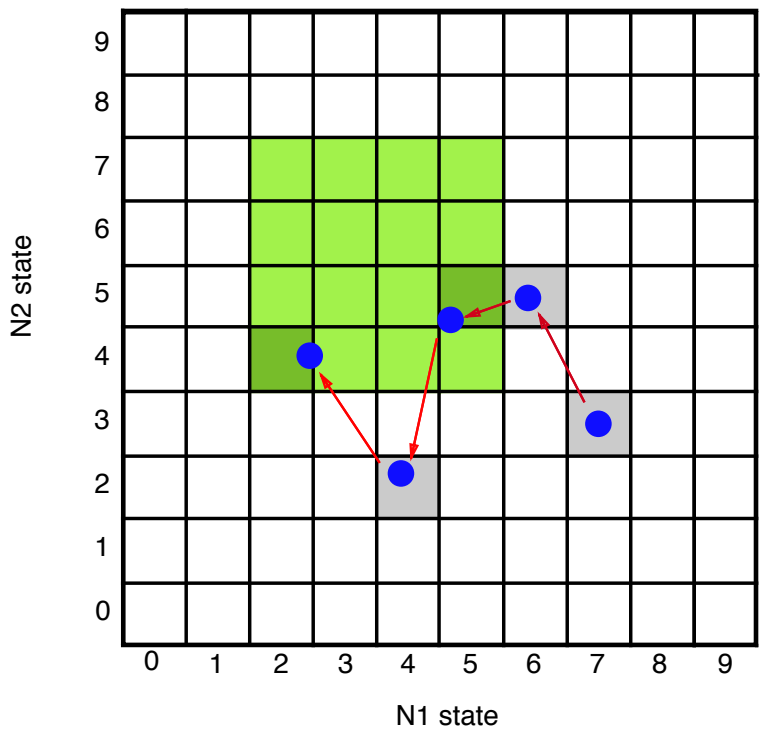

Figure 1: Representation of how an agent travels through a two dimensional state space by taking actions (red arrows) with the aim of fulfilling the reward condition (staying inside the green region). In our application the states represent discretised populations of two species of bacteria, $\mathbf{N}_1$ and $\mathbf{N}_2$. The actions taken are changing auxotrophic nutrient concentrations and a reward is given if the populations are within the specified range.

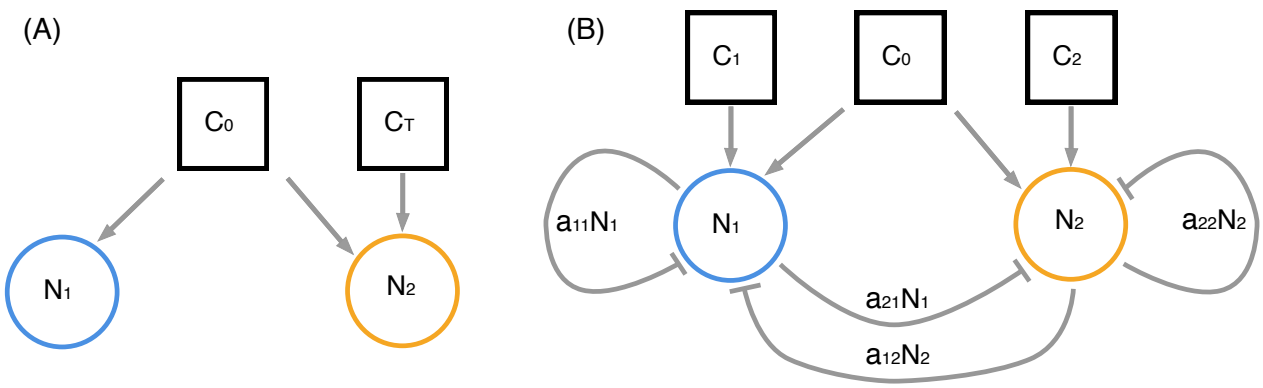

Figure 2: Sketches of the two main systems. (A) The single auxotroph system with no Lotka-Volterra competition, two species grow in a chemostat. Both require the common carbon source C_0. Only population 2 requires tryptophan C_T. (B) System of two auxotrophs dependent on two different amino acids, with inhibitory Lotka-Volterra interactions

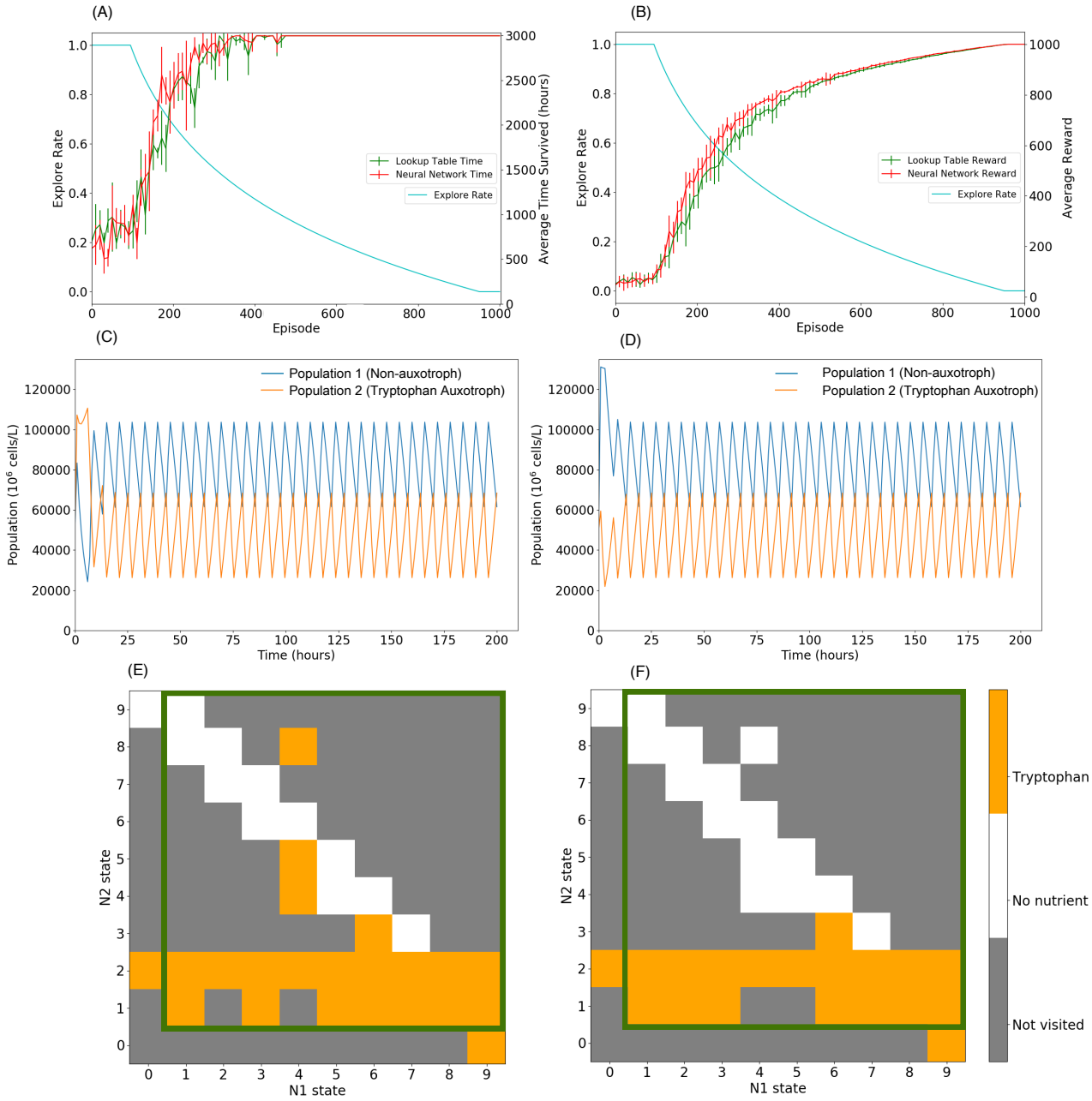

Figure 3: **Single auxotroph system**. (A) Moving average of the number of time steps survived as training progressed and (B) moving average of the total reward received per episode survived as training progressed (100 point average). Both are averaged over ten training runs and error bars represent one standard deviation. (C, D) Time-course of 200 time steps under control of the trained LT and NN agents respectively. (E, F) State-action plots, in the discretised state space, of the LT agent and NN agent respectively. An orange square represents adding the auxotrophic nutrient, a white square represents adding no nutrients, the greyed out states were never visited. The green box represents the area of state space in which the agent receives a positive reward. Here the agents can distinguish populations up to 120000×10^6.

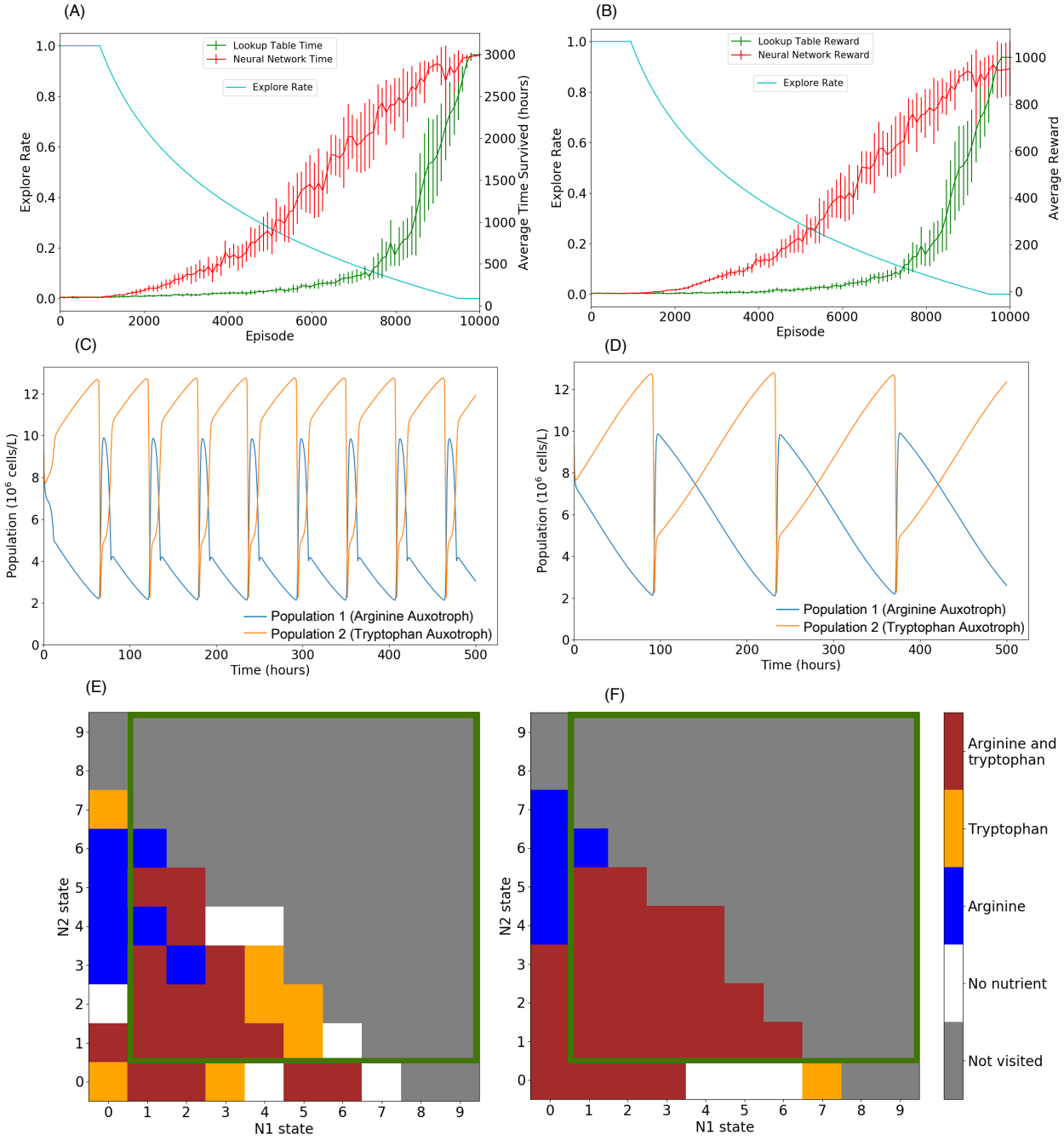

Figure 4: **Double auxotroph system with coexistence**. (A) Moving average of the number of time steps survived as training progressed and (B) moving average of the total reward received per episode survived as training progressed (100 point average). Both are averaged over ten training runs and error bars represent one standard deviation. (C, D) Time-course of 500 time steps under control of the trained LT and NN agents respectively. (E, F) State-action plots, in the discretised state space, of the LT agent and NN agent respectively. The green box represents the area in state space where the agent receives a positive reward. Here the agents can distinguish populations up to 20×10^6.

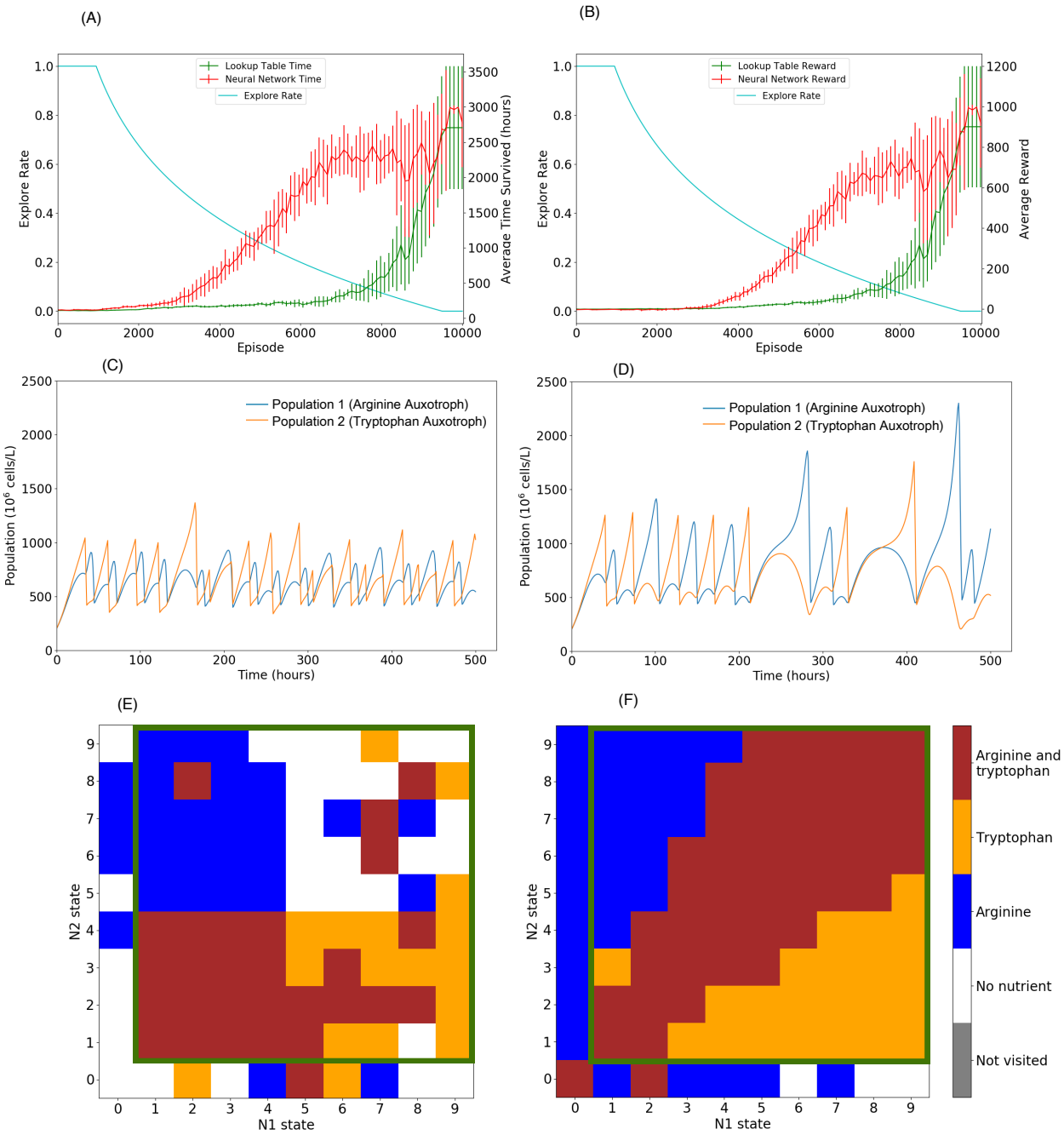

Figure 5: **Double auxotroph system with no coexistence**. (A) Moving average of the number of time steps survived as training progressed and (B) moving average of the total reward received per episode survived as training progressed (100 point average). Both are averaged over ten training runs and error bars represent one standard deviation. (C, D) Time-course of 500 time steps under control of the trained LT and NN agents respectively. (E, F) State-action plots in the discretised state space, of the LT agent and NN agent respectively. The green box represents the area of state space in which the agent receives a positive reward. Here the agents can distinguish populations up to 1000×10^6.

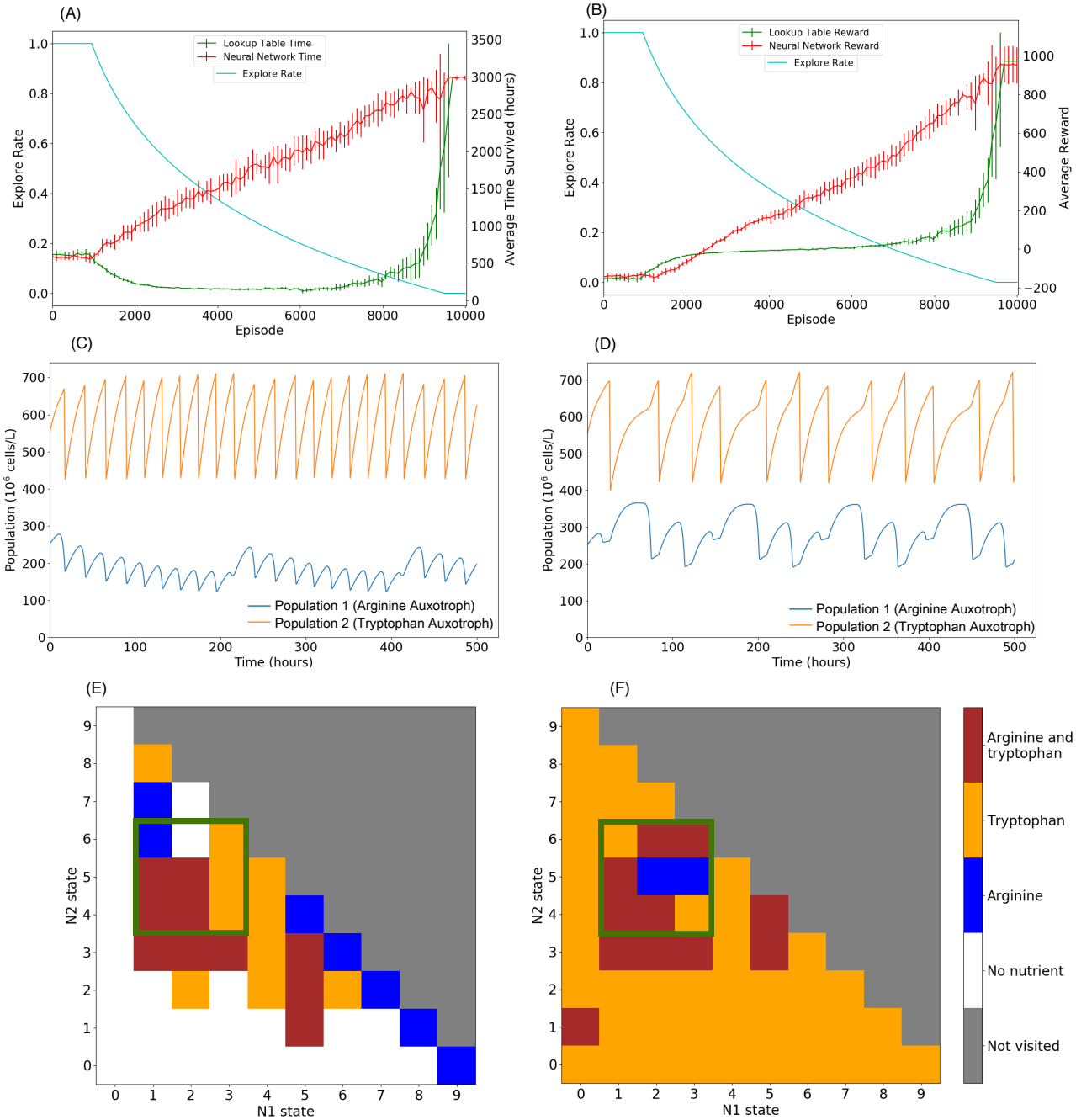

Figure 6: **Double auxotroph system with a smaller reward target**. (A) Moving average of the number of time steps survived as training progressed and (B) moving average of the total reward received per episode survived as training progressed (100 point average). Both are averaged over ten training runs and error bars represent one standard deviation. (C, D) Time-course of 500 time steps under control of the trained LT and NN agents respectively. (E, F) State-action plots in the discretised state space, of the LT agent and NN agent respectively. The green box represents the area of state space in which the agent receives a positive reward. Here the agents can distinguish populations up to 1000×10^6.

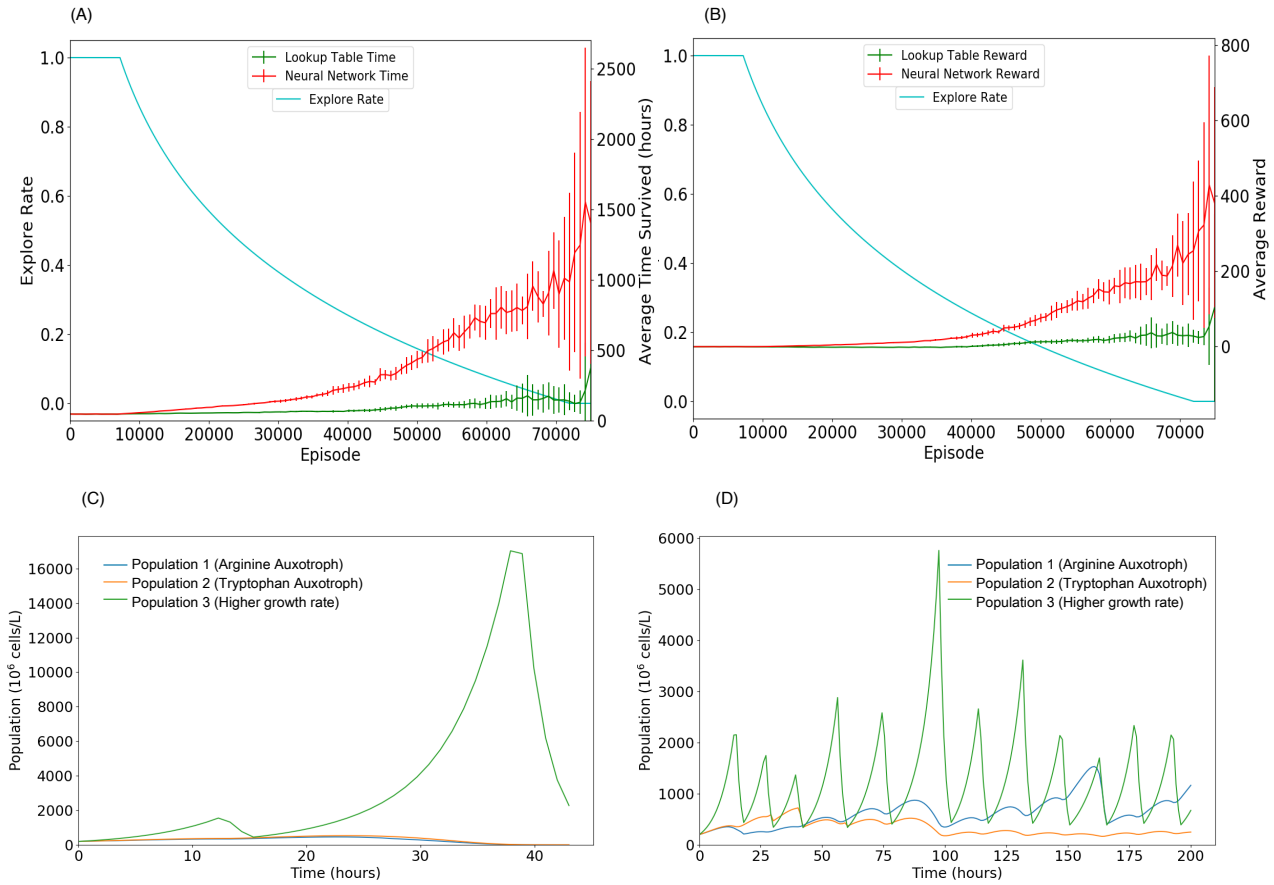

Figure 7: **Triple auxotroph system**. A) Moving average of the number of time steps survived as training progressed and (B) moving average of the total reward received per episode survived as training progressed (750 point average). Both are averaged over ten training runs and error bars represent one standard deviation. (C) Time-course of 109 time steps under control of the trained LT agent. (D) Time-course of 200 time steps under control of the trained NN agent.