# Learning to control coupled dynamical systems


**Devika Subramanian** DEVIKA@RICE.EDU
Department of Computer Science, Rice University, Houston TX 77005

**R. Peter Bonasso** R.P.BONASSO@JSC.NASA.GOV
NASA Johnson Space, Center ER2, Houston TX 77058

**David Kortenkamp** KORTENKAMP@JSC.NASA.GOV
NASA Johnson Space, Center ER2, Houston TX 77058



## Abstract

Coupled dynamical systems are an interesting, commonly-occurring class of systems with the property that the behavior of their subsystems is deterministic and easily derivable in isolation; yet it is impossible to analytically predict the behavior of the overall system. Such systems are known to be sensitive to initial conditions and are typically studied using numerical simulations. In this paper, we experimentally study machine learning techniques to control an advanced life support system, a representative example of a coupled dynamical system. It is difficult to design tractable formulations of reinforcement learning to acquire closed-loop control policies for such a system because we have no *a priori* knowledge of system dynamics. We present a two-step method for acquiring good control policies — we first learn a "short" open loop control plan that is applied *repeatedly* to maximize the given control system objective. This open loop plan provides insight into the topology of the system state space, which guides the design of a reduced state and action space, as well as an informative local reward function. Armed with this new and effective formulation, a Q learner finally acquires an optimal closed-loop control policy for the system.


## 1. Introduction

Coupled dynamical systems are an interesting, commonly-occurring class of systems with the property that the behavior of their subsystems is deterministic and easily derivable in isolation; yet it is impossible to analytically predict the behavior of the overall system. Such systems are known to be sensitive to initial conditions and are typically studied using numerical simulations. The control of coupled dynamical systems is an open problem. In this paper, we experimentally study machine learning techniques to control an advanced life support system (ALS), a representative example of a coupled dynamical system. Our goal is to acquire an optimal closed-loop control policy for the system that maximizes an externally specified objective function.

A "generative model" [4] of the coupled dynamical system is available, and we attempt to determine an optimal control policy by appropriately sampling state trajectories using reinforcement learning. Let the coupled dynamical system $C = (S, A, next, r)$ have state space $S$, action set $A$, state transition function $next : S \times A \rightarrow S$, and local reward $r : S \times A \rightarrow \Re$. The control objective is to maximize the undiscounted sum of local rewards over the (finite) lifetime of the system: $\sum_{t\geq 0}^{N}[r(s_t, a_t)]$. The state and action spaces are products of the state and action spaces of the component subsystems and in addition include system-level state variables and actions. The transition function *next* and local reward function $r$ are not available explicitly, but can be sampled using the generative model. Further, for the systems we consider here, $r$ is not informative; there is a delay of several hundred decision-making steps before a useful reward signal is provided. If $C$ is fully observable or can be made so by state augmentation, we can treat it as a Markov decision process and learn an optimal value function $V^* : S \rightarrow \Re$ for the system using reinforcement learning [6]. The value function $V^*$ maps states to the optimal expected sum of rewards from that state, and it indirectly specifies an optimal closed-loop control policy $\pi^* : S \rightarrow A$ mapping states to actions. The standard approach to designing reinforcement learning algorithms is to manually construct tractable state and action spaces [2, 7], and to manually devise good progress measures [7] to compensate for the lack of an informative local reward function. Performing these two steps requires knowledge of system dynamics that is good enough to identify key state space variables and actions. If the required state and action spaces are too large to be represented exactly, there are methods to learn approximate value functions [7, 2]. However, few convergence guarantees are available for algorithms that learn such approximations and deep system knowledge is required to posit good functional forms for the approximate value functions.

We present a two-step method for acquiring good control policies for dissipative coupled dynamical systems, i.e., ones that have finite lifetimes. The ALS is dissipative because it eventually run out of resources for every control policy and enters a terminal state. The control goal is to maximize the undiscounted sum of rewards over that indefinite but finite horizon. In the first step, we learn a "short" open loop control plan that is applied *repeatedly* to maximize the given control system objective. The open loop plan is a sequence of actions from $A$. To find such an action sequence, we do not need to make the Markov assumption of full observability of the state space $S$ or attempt to construct abstractions or approximations of $S$. We work directly with $A$ to find a short sequence that when applied repeatedly, maximizes the externally specified objective function.

This open loop plan provides insight into the topology of the system state space. It identifies the system as having a small core of "safe" states. A safe system state is one in which all component subystems are functioning normally. To maximize the objective function it is necessary to keep the system in the safe core for as long as possible. The repeated execution of the short sequence of actions keeps the system state in a periodic trajectory through the safe core for as long as it is feasible.

In the second step, we analyze the system trajectories generated by the repeated execution of the short open loop plan. This analysis directly yields the important state and action components leading to small, explicitly representable state and actions sets. It also suggests the construction of an informative local reward function. Armed with this new and effective formulation, a Q learner finally acquires an optimal closed-loop control policy for the system. As such, we present the beginnings of a methodology for using machine learning techniques as "probes" in understanding dissipative coupled dynamical systems. This methodology has implications both for control system designers and for machine learning researchers.

The rest of the paper is organized as follows. In Section 2, we briefly describe the advanced life support system which is the focus of our experimental study. We discuss the difficulties with learning a closed-loop control policy for the system and demonstrate the pitfalls of manual state abstraction without some knowledge of the dynamics of the overall system. We present the results of our two-step method for acquiring an optimal closed-loop control policy for this system. In Section 3, we conclude with a discussion of the significance of our results and present open questions that our work raises for machine learning research.

## 2. Learning to control an advanced life support system

An ALS system which regenerates much of its water, air and food is critical for long-duration space missions, since it minimizes both the stores of consumables and the need for resupply. Since ALS systems have small buffers and low margins, optimal control is essential to their success. However, tight interactions among ALS components and the lack of models of system-level behavior make hand-crafting an optimal controller difficult. It therefore forms an ideal test bed to study algorithms for automatically acquiring control policies for coupled dynamical systems.

### 2.1 A simple ALS simulation

A life support system for a crew on a planetary surface includes the processes of air revitalization, water recovery, food production via grain crops and the biological processes of crew in their habitat as they carry out their missions. The crew consumes clean air, water and food and produces waste water, exhalant air and science, a general term for the crew's productive output. All of the processes except the crew require energy.

Based on a number of past efforts with controlling life support systems [1] as well as discussions with NASA's BIO-Plex engineers we developed a deterministic simulation (see Figure 1) of an abstraction of these four processes. The simulation includes stores for energy, clean air, clean water and food. The control choices for the simulation are the allocation of energy per simulation unit time to the various processes or modules, and whether to draw from the various stores at any given time step. A last control choice is the activity level of the crew: higher levels consume more air, water and

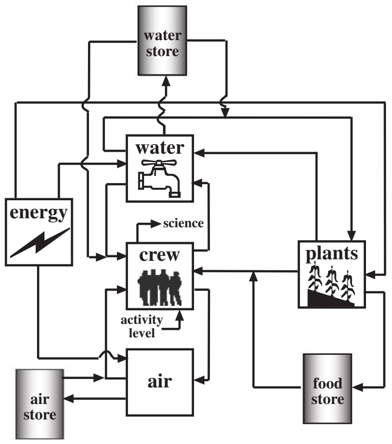


*Figure 1.* A simple ALS simulation and initial values.

food but produce more science.

Simulation time is measured in 'ticks', which are nominally one hour. The ALS system state space is the product of the state spaces of the four modules (internal states, inputs and outputs and states of the associated stores), and system level variables: the total energy left and the total simulation time elapsed. The ALS action space is the product of the action choices for each of the modules: energy allocations to each module, whether or not to use a corresponding store, and the crew activity level.

In a single tick each module makes a state transition based on the relevant component of the system action vector. The state transitions of the four modules (along with updates of global state variables) yields a one-step ALS system transition. The crew module takes in clean air, clean water and food and produces waste air, waste water and science. The air recovery process takes in energy and waste air from the crew and produces clean air. The water recovery process takes in energy and waste water from the crew and plant crops and produces clean water. The crop module takes in energy and a portion of the clean water and produces food for the crew and waste water. If a store is to be used on a given tick, no energy is consumed for that module during that tick. Since all processes are degenerative (i.e., not mass-preserving), the simulation will eventually terminate due to air, water or food losses. Observable output from the simulation includes the amount of waste air and waste water from the crew module at each tick, the amount of science produced by the crew at each tick, the cumulative science and the total elapsed time since the start of the simulation.

The ALS system is a hybrid coupled dynamical system with its state and action spaces composed of a mixture of continuous and discrete variables. Each subsystem of the ALS is a deterministic finite state machine since each represents a physical process with inertia: it takes time to get to steady state operation, as well as to degrade in the absence of energy. The water and air recycling processes can enter periods of indefinite hibernation in the absence of energy, while the crew and crop modules shut down terminally under conditions hostile for life. The ALS system is thus composed of four interacting deterministic finite state machines. While we have detailed models for each of the four processes, there is no explicit model of the entire system. The evolution of the states of the entire system is dependent on the starting conditions. In short, the interactions among the subsystems are sufficiently complex that we cannot derive them analytically. The optimal control policy for such a system is a function of the desired objective function; in this paper we consider maximizing a linearly weighted function of mission duration and science output from the crew.

### 2.2 Learning closed loop policies using reinforcement learning

The ALS system appears to be well-suited for the application of reinforcement learning to acquire a closed-loop policy for optimal control. However, the sizes of the state and action spaces and the lack of an informative local reward function makes it impossible to use the full formulation above with any existing reinforcement learning algorithm.

As a prelude to understanding aspects of the ALS system state that are important to action choice, we use reinforcement learning to acquire the best policy possible with extremely limited state information. We compare it against a baseline policy that uses the same limited state information. The baseline policy can be stated simply as *use air, water and food stores up before depending on the air, water and food production processes*. We conjecture that using food, air and water reserves first maximizes science output,q and stretches the energy reserves out furthest (maximizing lifetime) since using the stores requires no energy.

With an initial crew of four, 100 initial stored units each of air and food, 200 units of stored water and 1000 units of energy and an allocation of four units of energy to each process, the baseline policy produces a mission duration (activity level = 1) of 56 ticks (and 95 units of science) or a cumulative science (activity level = 3) of 246 units (in 44 ticks). The air and water stores are depleted in 24 ticks in both cases and there are over 700 units of energy left. To conserve space, we will concentrate on the objective of maximizing the cumulative science for the remainder of the paper.

Now we ask the question: how well does an optimal controller, acquired by a value function based reinforcement leaner with state information that includes stores, compare to the simple controller sketched above? We use Q learning [8], a temporal difference (TD) reinforcement learning algorithm. The Q learning algorithm learns the $Q^* : S \times A \rightarrow \Re$ function which maps each state-action pair in the product of the state and action space to a real number which is the expected sum of rewards for taking that action in that state. We now design a state-action space that preserves information about the stores and which keeps the size of the space under $10^4$, to obviate the need for function approximation.

If we use the store levels as part of the state information: we are faced with a state space size of $100*100*200*1000 = 2\times10^9$ assuming a discretization of 1 on the stored quantities[1]. Unfortunately, store levels are not directly visible from the simulation output. So we use the previous action vector with information about whether or not stores were used in the last cycle as an indication of whether stores are available or not. This is not an entirely accurate signal, but in conjunction with energy allocated to the different modules, it correlates highly with actual use of stores. So, the baseline policy is representable in our state space choice of the previous action vector for the system. The size of our state space is the product of 8 (whether or not each of 3 stores were used), 3 (the three crew activity levels), and 27 (energy allocated to the air, water and crop modules considering only the values 3, 4 and 5 for energy allocations per tick)[2]. If we used an action space with the same cardinality, we would have about 420,000 state-action pairs, far above our goal of about 10,000 state-action pairs. Therefore, we reduce the action space into two components: a quantity to be changed (energy allocation to air, water and crops, activity level, air, water, and food stores), and a change to the quantity (no change, increase or decrease by one). This conceptualization brings the action space size to 21. Now our Q table size is about 13600. Finally, we used the following local reward function:

$$r(s,a,s') = \begin{cases} -10000 & \text{if } s' \text{ is terminal} \\ science(s) & \text{otherwise} \end{cases}$$

We use a standard zero-initialized Q-learning[8] algorithm with 0.1 epsilon-greedy sampling, a learning rate of 0.9, and no discounting. The primary result we obtain is the sensitivity of the value of the final policy to the starting state of the simulation. For example, the final learned policy produces 185 units of science in 38 ticks before mission end in one experiment, and 98 units of science in 23 ticks in another experiment. To reduce the variance on the quality of the final result, we start from a benign nominal starting state in subsequent experiments. While this change reduces policy value variance, we notice policy instability: convergence to wildly different policies with similar values during different runs. We do not obtain the use stores first baseline policy, perhaps because it was not optimal for the given objective function of maximizing cumulative science output. However, it should be noted that we obtain 322 units of science in 20 ticks, while the baseline policy produced less science (246 units) over a longer time period (44 ticks).

This experiment shows that we need a richer representation of the state space to be able to produce better science output, as well as a better local reward function. However, the addition of any left out state variable increases state space sizes significantly, and we have no basis for preferring the inclusion of one state variable over another. We note that a direct search in policy space suffers from the same limitations because of the need to reduce the state and action space sizes.

We now present the results of our two-step method for learning an optimal closed-loop control policy for the system. First, we describe the results of our search for a short open loop plan that is executed repeatedly.

### 2.3 Searching for open loop plans using genetic algorithms

We use the GAL implementation [5] of a genetic algorithm to search the space of open loop plans of length less than $20^3$ The system action vector is encoded as a bit string as follows: 2 bits each for the energy allocations (0, 1 and 2 indicate 3, 4 and 5 energy units respectively, and 3 = 5 as well), 2 bits for the activity level (1,2, and 3, and 0 = 1 as well), and 1 bit for each of the stores (1 = store to be used, 0 = store not to be used) for a total of 11 bits (see Figure 2). To

[1] We cannot choose a bigger discretization on these quantities without knowledge of the system

[2] This choice was not arbitrary. The processes nominally need 4 units of energy per tick and we looked at values one above and one below.

[3] This choice is made based on setting a cut off time of a couple of hours for the operation of the GA.

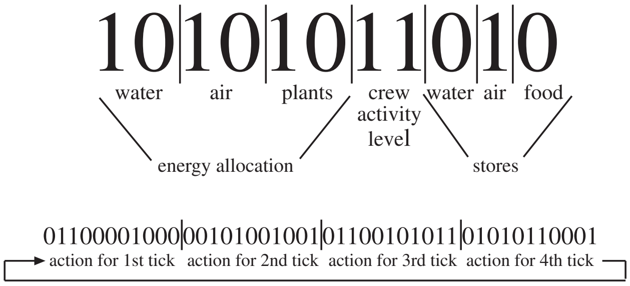


*Figure 2.* At top is the original genetic algorithm bit string representing actions to be taken. At bottom is a multi-step bit string of length 4.

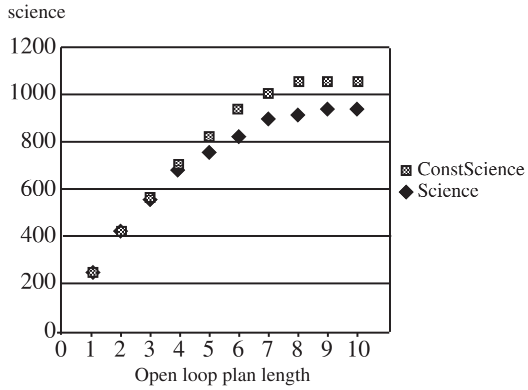


*Figure 3.* Quality of open loop plan as a function of plan length.

construct plans of length $n$, we repeated the string $n$ times (see Figure 2). The GA parameters used are: two point crossover maintained at 60%, a low mutation rate, selection based on proportional fitness, and population size of 100. The evaluation function is the cube of the cumulative science output[4]. To evaluate a string, the simulation is run using that string for $n$ ticks (where $n$ is the length of the plan), and repeating it every $n$ ticks until the mission ends. We typically run twenty experiments and choose the best string among the strings with the highest value from each experiment. The best single step open loop plan obtained by this technique produces 184 units of science in 26 ticks; far below the values of the baseline policy and the policies produced by the Q learner. We performed experiments with open loop plans of length $n$ starting at two, and use the value of the best individual from those experiments as the value to beat for $n+1$, continuing to increase $n$, the number of actions in the sequence, until the next increase resulted in no improvement.

As shown in Figure 3 the results are fairly spectacular. The diamonds in Figure 3 show a plot of the best cumulative science achieved versus increasing plan size. Surprisingly, the quality of the open loop plans levels off after a certain length. The GA achieves 938 units of science (in a 120 tick mission duration) with a nine step (99 bit string) open loop plan. An examination of the plan shows that it tends to keep the activity level at three, and uses the lowest energy level – three units – for each process. With this insight, we repeated the experiments with the activity level set to 3 and the energy allocation set to 3 throughout. The plot of squares in Figure 3 shows that the GA finds the best open plan to be of length 8 rather than length 9, with a higher science solution (1049 units as opposed to 938).

[4]This choice was made to be able to better differentiate the quality of open loop plans generated.

### 2.4 State space analysis

We analyze the behavior of the eight step plan executed repeatedly to gain an understanding of its effectiveness. As discussed above, the GA helps fix the values of several key system parameters: the energy allocations for each subsystem needs to be three units per tick, and maintaining an activity level of 3 for the crew is necessary for maximizing science output. Figure 4 shows the results of running the eight step open-loop plan for the first 100 ticks of the simulation. Plotted against ticks are the crew output of waste air (air-dirty or AD), waste water (water-dirty or WD) and science as well as the store allocation decisions (WS and AS for water and air stores respectively). What is clearly evident is that the plan lets the output of the air and water processes decay for a certain period of time before resuscitating them by using the stores. This pulsing action spreads the stores out over the course of the mission, thus increasing the number of ticks in which to accumulate science, while maximizing the average science per tick. Additionally, the plan uses the plant stores on every tick.

The decay curves of the air and water are a direct result of the decay laws of the air and water processes in the simulation, assuming no use of stores. The air available to the crew at step $n$ is given by: $air_n = ncrew \times (aclf)^n$, where $ncrew = 4$ (number of crew), and $aclf$ is the air loss across the crew process and is set by the activity level. For example $aclf = 0.95$ for activity level 3. For this activity level, the air at step 8 before a store use is 2.65 as shown in Figure 4.

The water available to the crew at tick $n$ is also a power function. It is half the total water coming from the water recovery process, since we allocate that output evenly between the crew and the crops. The dirty water input to water recovery on step $n$ is the sum of the crew dirty water output and the plant dirty water output from the previous tick. The crew dirty water output is in turn a function of the clean water to the crew from the previous tick times the water crew loss factor ($wclf$). Such a recursive computation yields a sum of powers in $n$ of the crew loss factor. $WTC_n = (C_{0,n}WD_{plants,0} + C_{1,n}wclf + C_{2,n}wclf^2 + ... + C_{n,n}wclf^n)(R/2)^n$, where $WTC_0 = WD_{plants,0} = C_{0,1} = ncrew$, $C_{1,1} = WTC_0$, $R = 0.97$ (water recovery factor), and $C_{i,n} = C_{i-1,n-1} + C_{i,n-1}$. At $n = 8$, this recursive formula predicts the water from the crew will be 1.07 as shown in Figure 4.

The use of the plants stores on every tick conserves energy. Based on the food requirements of the crew, the food stores would last over five hundred ticks, far outlasting the air and water stores. The net result of the 8-step open loop plan's periodic use of stores was to keep the science output above four units at all times. The success of the repeated execution pf the short open loop plan found by the GA suggests the following key properties of the state space of the ALS.

- A very small fraction of the entire state space is "safe". A system state is safe if all the component subsystems are functioning normally (rather than being in hibernating or dead states). The objective function (undiscounted sum of rewards) is maximized if the system can be made to stay in this safe core of system states as long as possible. The start states of the system must be in this safe core.
- The "diameter" of the safe core of the state space is small. By "diameter," we mean the average number of random actions that will take the system away from the safe states. Random walk experiments with the simulation confirm this property.
- To minimize resource consumption and maximize science, it is best to keep the system in a periodic trajectory through the safe states for as long as possible. Such a policy runs consumable resources in the individual subsystems down to critical levels and then restores them in a single pulse. This keeps the number of "refills" down (since there is a cost to refilling) and keeps each subsystem safe. An apt analogy is the strategy of keeping a balloon afloat by periodic upward thrusts to counter the effects of gravity.

Our analysis suggests that the important aspect of the control policy to be learned are the critical levels of each resource to be maintained at all times which in turn determines the periodicity of the trajectory in the safe core of states and the timing of the use of the stores. For this simulation, these critical resource levels are highly correlated with the outputs of the crew module. This is the key contribution of the simpler search in open loop plan space. It suggests the new variables that should form the state space for the search of a closed loop plan either by value function or direct policy search methods. We point out that none of the current learning methods have automated ways of constructing such new variables for expressing control policies.

### 2.5 From open loop back to handcrafted closed loop control

Based on these findings we hand-craft a closed loop controller as follows. At each tick, we allocate three units of energy to each process and use the plant stores. If the science output for the tick is below 4.0, we use the air and water stores. If the science output is at or above 4.0, and the air output from the crew is at or above 2.65, but the water output from the crew is below 1.0, we use the water stores. If the science output is at or above 4.0, and the water output from the crew is at or above 1.0, but the air output from the crew is below 2.65, we use the air stores. Otherwise, we do not use the air or water stores. The parametric structure of this policy as well as the constants is revealed by the analysis of the eight step open loop plan presented in the previous section.

The resulting policy produces 1051 units of science (2 units more than the open loop plan), over a mission duration of 198 ticks (the same mission duration for the best open loop plan for maximizing mission duration.) Our closed loop controller performs better than the open loop plan toward the very end of the mission duration where the air and water crew outputs vary irregularly rather than with the eight tick periodicity, and thus the closed loop policy is able to change its store use pattern to match the situation. Thus, guided by the GA we are able to achieve a little more science and match the mission duration of the best open loop GA generated plan, with our hand-crafted closed loop controller.

### 2.6 Automatic generation of closed loop control policy

We can now design significantly reduced state and action spaces and a more informative progress measure for a reinforcement learning algorithm. The state space consists of water, air and science outputs of the

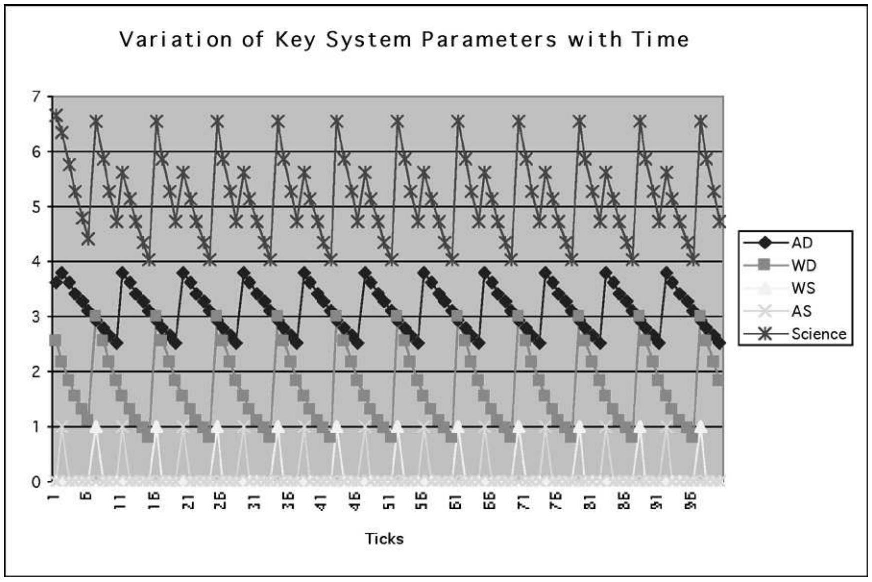


*Figure 4.* Analysis of open loop 8-step plan.

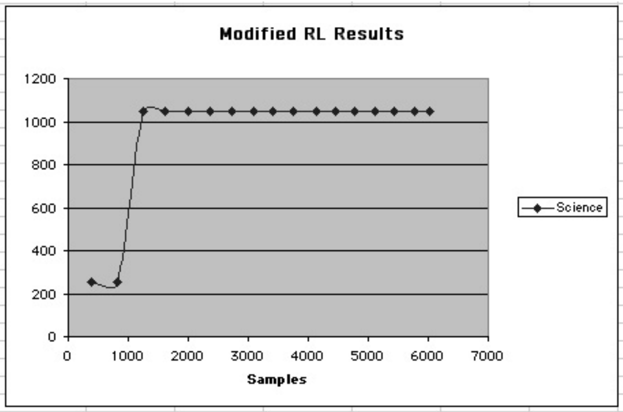


*Figure 5.* The learning curve for Q learning.

crew. The continuous water and air outputs are discretized[5] in intervals of 0.5 over the range $[0, 4]$ and the continuous science output is discretized in intervals of 1 over the range $[0, 5]$. This yields a state space size of $9 * 9 * 6 = 486$. The action space consists of the decision to use air and and water stores and has size 4. The food store is used throughout the simulation run, the crew activity level is set to 3 to maximize science output, and all subsystems get an energy allocation of 3 units per tick. The size of the $Q$ matrix is $486 * 4 = 1944$. We shape the behavior of the learned closed-loop control policy by rewarding the system for keeping science above 4 units and for using the air and water stores at a rate close to that used by the repeated execution of the open loop plan. The specific local reward function is a linear combination of three terms: the difference in the discrete derivatives of science, air and water outputs compared to that of the open loop control plan found by the GA.

$$
\begin{aligned}
r(s_t, a_t) &= 0.5 * sf(s_t) + 0.75 * af(s_t) + 1.0 * wf(s_t) \\
sf(s) &= \begin{cases} sci(s_t) & \text{if } sci(s_t) > 4.0 \\ sci(s_t) - sci(s_{t-1}) & \text{o/w} \end{cases} \\
af(s_t) &= -abs(GA\_adrv - (air(s_t) - air(s_{t-1}))) \\
wf(s_t) &= -abs(GA\_wdrv - (wtr(s_t) - wtr(s_{t-1})))
\end{aligned}
$$

Since the system is dissipative and all trajectories end in a terminal state with a dead crew, we provide the above reward as long as the crew is alive in $s_t$ or if the crew dies after performing a given level of cumulative science. Otherwise, a high negative reward is given to terminate further exploration of such paths by Q learning.

We use a variant of Q-learning developed in [3] in which all entries are initialzed to the highest possible values they can attain. This obviates the need for ad-hoc setting of the exploration policy. An action choice policy that picks the action with the highest Q value in a state s ensures that every action is tried at least once from every state. Further, since there are repeated states in each simulation run, we ensure that a state is not penalized multiple times per episode. Since all of the trajectories in the system end in "death", we distinguish between good deaths when high levels

[5] The selection of discretization of these continuous variable is outside the scope of this paper.

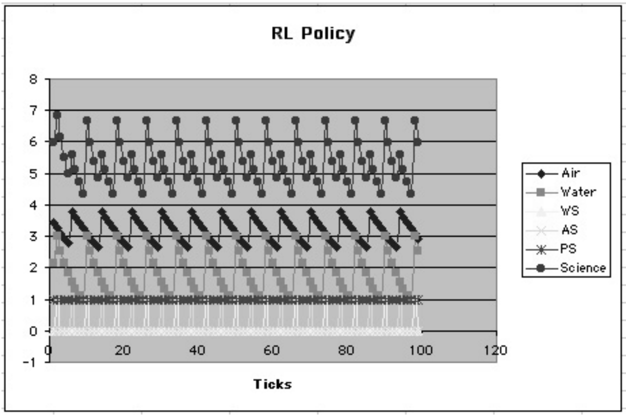


*Figure 6.* Analysis of closed loop control policy acquired by Q learning.

of science has been achieved from bad deaths when the cumulative science output is low. During learning, we choose an action in state $s$ that maximizes $Q(s,a) * good\_deaths(s)/total_{d}eaths(s)$. This action selection biasing technique is taken from [3]. The learning curve of the Q learner is shown in Figure 5. The behavior of the optimal policy derived by the Q learner is shown in Figure 6.

The optimal closed loop control policy derived achieves 1049.5 units of science in 198 ticks. Note that the hand crafted controller obtains 1051 science units in 198 ticks. We attribute this difference to the coarseness of the discretization of the air output which is sampled at granularity 0.5 by the Q learner and has value 2.65 for the hand designed controller.

## 3. Discussion and Conclusions

The solution space for a closed loop control policy in dissipative coupled dynamical systems such as the ALS system is enormous. In this paper, we demonstrate the effectiveness of machine learning techniques in "probing" the state space to provide control system designers useful characteristics of its topology. In this way, machine learning algorithms become not just a tool for adjusting an on-line control system, but also a tool for helping programmers design an *a priori* control policy. There are some important research questions that need to be answered to further the two-step approach for system analysis and control advocated here 1) For what other classes of dynamical systems is open loop plan analysis helpful for guiding state space design for closed loop policy search? 2) How do we design reinforcement learning algorithms that can construct the right state space features by automating the meta-analysis we demonstrated in this paper? The performance analysis of the open loop eight step plan was key to uncovering the structure of the state space. How can we automate such performance analyses? 3) What are other machine learning algorithms that can effectively learn useful properties of large state spaces?

## References

[1] S. Crawford, C. Pawlowski, and C. Finn. Power management in regenerative life support systems using market-based control. In *Proceedings of the 30th International Conference on Environmental Systems*, 2000.

[2] R. H. Crites and A. G. Barto. Improving elevator performance using reinforcement learning. In *Neural Information Processing Systems*, pages 1017–1023, 1995.

[3] S. Griffin and D. Subramanian. Reinforcement learning in a high-dimensional state space. Technical report, Rice University, 1998.

[4] M. Kearns, Y. Mansour, and A. Ng. Sparse sampling methods for planning and learning in large and partially observable Markov Decision Processes. In *Proceedings of the International Conference on Machine Learning*, 2000.

[5] W. M. Spears. Gal: Genetic algorithms in lisp. Technical Report http://www.aic.nrl.navy.mil/spears/freeware.html, Navy Research Center for Artificial Intelligence, 2000.

[6] R. Sutton and A. G. Barto. *Reinforcement learning: an introduction*. MIT Press, 1998.

[7] G. Tesauro. Temporal difference learning and TD-Gammon. *Communications of the ACM*, 38:58–68, 1995.

[8] C. J. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.