Analytical Design of Reinforcement Learning Tasks

Robert E. Smith

Department of Aerospace Engineering and Mechanics
The University of Alabama
currently on sabbatical at
The Intelligent Computing Systems Centre
The University of The West of England
Bristol, UK
email: rsmith@btc.uwe.ac.uk

Abstract

Reinforcement learning (RL) problems constitute an important class of learning and control problems faced by artificial intelligence systems. In these problems, one is faced with the task of providing control signals that maximize some measure of performance, usually taken over time, given feedback that is not in terms of the control signals themselves. This feedback is often called "reward" or "punishment." However, these tasks have a direct relationship to engineering control, as well as the more cognitive intelligence related areas suggested by these terms (Barto, 1990).

In recent years, many algorithms for RL have been suggested and refined. Notable are those discussed by Sutton, Barto, and Watkins (1989), Holland's Bucket Brigade (Holland et al., 1986), Watkin's Q-learning algorithm (Watkins and Dayan, 1992), and others. Despite these advances, there remains no standard, analytical methods or test suites for empirically evaluating reinforcement learning systems. Most test problems in RL are in abstract, difficult to analyze forms (e.g., maze problems, the inverted pendulum, etc.).

This paper describes a method for designing arbitrary numbers of RL test problems with clear, parameterizable characteristics. An additional contribution of this method is a clear identification of parameters for RL environment complexity. Although the method described is primarily suggested for RL problem design and RL method evaluation, its parameters could be used for analysis of extant RL test problems.

The following sections outline basic RL nomenclature, followed by the development of Walsh transform based design techniques for a limited class of RL problems. Straight forward extensions of these techniques to a broad class of RL problems are also provided. Use of these techniques are further extensions are discussed.

Introductory Assumptions and RL Nomenclature

To begin, consider deterministic, finite horizon, un-discounted, binary decision tasks. That is, at each time step *t*, the plant being controlled is in one of a finite number of states, *s(t)*. The environment provides a signal accurately indicating the system's state. The system must submit a binary action signal at (right or left, right or wrong, -1 or +1). Once this signal is submitted, the system makes a deterministic transition to another state, *s(t+1)*. The environment provides a reward signal, *r(t,s(t),a(t))*. Assume that this process goes on for a known, maximum number of time steps *L*. To illustrate environments, we will use diagrams like that shown in Table 1. Each cell in this table represents a state of the environment, and the digits in the cell represent the action sequence that leads to that state. Note that going to the left of right below any given cell adds the appropriate action bit to the action sequence.

Time Step	Action Sequences *a* (right = -1 action, left = +1 action)							
0	(null, this is the starting state)							
1	+1				-1			
2	+1+1		-1+1		+1-1		-1-1	
3	+1 +1 +1	-1 +1 +1	+1 -1 +1	-1 -1 +1	+1 +1 -1	-1 +1 -1	+1 -1 -1	-1 -1 -1

Table 1

Although *L*=3 examples will be used in this paper's tables for clarity and brevity, all techniques discussed apply to problems of more meaningful length.

[1] Copyright © 1998, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

Assume that the total value of the system's performance is simply the (undiscounted) sum of all the rewards,

$$R(a) = \sum_{t=1}^{L} r(t, s(t), a(t))$$

In following calculations we will refer to the vector of binary actions for *t=1…L* simply as *a*.
Note that we will later relax many of these assumptions, notably the assumptions of undiscounted reward, binary actions, and deterministic behavior. They are made here only to ease the initial Walsh transform calculations

Introduction to Walsh Transforms

The Walsh transform (Beauchamp, 1984; Goldberg, 1989a; 1989b) is analogous to the Fourier Transform. The Fourier Transform maps a continuous (or discrete) series of values spread over a continuous (or discrete) time domain to coefficients for various frequencies in that series. By analogy, the Walsh transform maps a discrete series of values spread over the domain of binary integers to coefficients for various sequencies in that series. The notion of a sequenecy is a bit less intuitive than that of a frequency, and requires some introduction. Sequencies can be thought of as partitions of the binary space. To distinguish partitions of this space from members of the space, we will describe partitions with the binary alphabet {0,1}, and members with the binary alphabet {-1,+1}. The partition 001 of the three bit binary space divides that space in two. Strings that have -1 in the third bit position, for instance -1-1-1, are in one half of the partition. Strings that have +1 in the third bit, for instance +1+1+1, are in the other half. Likewise, the partition 100 divides the space into two different halves. A two bit partition, like 101 divides the space into quarters, and so forth. We will refer to these partitions (or sequencies) by both their binary string representation (using the symbol *J*), and by their decimal interpretation as binary integers (using the symbol *j*). For instance, *J*=001 is partition *j*=1, *J*=100 is partition *j*=4, and *J*=101 is partition *j*=5.
To take the Walsh transform, assume we are given all the *R* values of a set of binary strings. An example is given in Table 2.

Binary String (*a*)	Value (*R(t,s,a)*)
-1-1-1	5
-1-1+1	-7
-1+1-1	15
-1+1+1	11
+1-1-1	13
+1-1+1	1
+1+1-1	-17
+1+1+1	-21

Table 2

In the Walsh transform, each partition *j* has a corresponding Walsh coefficient *w(j)* (which we will also call *w(J)*). Using the representation presented here, the Walsh coefficients are given by the following calculation:

$$w(j) = \frac{1}{2^L} \sum_{a \in \{-1,1\}^L} R(a) \prod_{i=1}^{2^L} \Psi\big(a(i), J(i)\big)$$

where *a(I)* is the *i*th bit of the action string *a*, and *J(I)* is the *i*th bit of the partition string, and *j=1,2,3…2L.* The four possible values of the function **Ψ** are shown in Table 3.

a_i	J_i	$\Psi(a_i, J_i)$
-1	0	+1
-1	1	-1
+1	0	+1
+1	1	+1

Table 3

Note that this function takes the value of the sign of the action bit, if that bit corresponds to a 1 bit in the partition string.
To illustrate the Walsh transform, Table 4 contains the Walsh coefficients for the reward values shown in Table 2.

J	*j*	*w(J)* or *w(j)*
000	0	0
001	1	4
010	2	3
011	3	2
100	4	6
101	5	0
110	6	-10
111	7	0

Table 4

To take the inverse Walsh transform, one simply sums the Walsh coefficients, sign being determined by the parity of the bit product of the corresponding partition and binary string:

$$R(t, s, a) = \sum_{j=0}^{2^L} w(j) \prod_{i=1}^{L} \Psi\big(a(i), J(i)\big)$$

Walsh Canonical Representation of RL Tasks

Once again restricting ourselves to deterministic, finite horizon, undiscounted, binary decision tasks, one can easily take the Walsh transform of *R*, the total value of the length L binary sequence of actions, at, *t = 0,1,2…L.*
Clearly, there are an infinite number of RL tasks of this type that would have the same Walsh coefficients. Therefore, it will be useful to introduce a Walsh canonical RL task representation. In this representation, the reward given at time step t is given by the Walsh sum for the first *t*-bit substring of the entire *L*-bit action sequence. This form

is depicted in Table 5. Note that each cell in this table is associated with a state shown in Table 1, and the signs within the cell represent the sign of the associated Walsh coefficients (*w(j)*) in the reward for time step *t* of the action sequence.

t	*j*	Sign of *w(j) in r*eward values *R*(*a(t)*)							
0	*0*	+							
1	*1*	-				+			
2	*2*	-		+		-		+	
	3	+		-		-		+	
3	*4*	-	+	-	+	-	+	-	+
	5	+	-	+	-	-	+	-	+
	6	+	-	-	+	+	-	-	+
	7	-	+	+	-	+	-	-	+

Table 5

Note that it parallels the procedure of the inverse Walsh transform. Rewards for the Walsh canonical form of the task shown in Table 2 are shown in Table 6. Note that a one-step "greedy" approach to this problem will yield a sub-optimal reward value of 13. If one uses one step look ahead, the optimal reward value of 15 can be obtained.

t	Reward values *R*(*a*) (right = -1 action, left = +1 action)							
0	0							
1	-4				4			
2	-1		1		-5		5	
3	-16	16	4	-4	-16	16	4	-4

Table 6

Interpreting the Walsh Coefficients

To interpret the Walsh coefficients in an RL context, one must first introduce two important parameters of the Walsh partitions. For this, we will borrow some terminology from genetic algorithms analysis (Goldberg, 1989c), which has extensively employed Walsh transform analysis.

First, one must consider the *order* of each partition. The order is the number of bits along which a partition divides the space. In the {1,0} alphabetic representation we have used, this is the number of 1 bits in the partition. For instance, partitions $j=1$ ($J=001$), $j=2$ ($J=010$), and $j=4$ ($J=100$) are all order one, and partition $j=7$ ($J=111$) is order three.

Second, one must consider the *defining length* of each partition. The defining length of a given partition is the maximum number of time steps between bits that divide the space. In our representation, this is the distance between outermost 1 bits in the partition. The order zero partition ($J=000$) and all the order one partitions (e.g., ($J=001$) and ($J=010$)) have defining length 0. Partition $j=7$ ($J=111$), as an example, has defining length 2, as does partition $j=5$ ($J=101$).

Given this terminology, one can interpret the Walsh coefficients as having clear meaning in RL tasks. The order of a partition represents the number of action decisions that influence a reward-gaining event. The defining length of a partition represents the delay in time of the influence of those actions. For instance, consider partition $j=7$ ($J=111$). The coefficient of this partition represents the joint reward-gaining effect of the actions at time steps 1, 2, and 3. As another example, consider partition $j=6$ ($J=101$). Its coefficient represents the joint reward gaining effect of actions at time steps 1 and 3.

To see how order and can be used to interpret the complexity of a RL task, once again consider the task shown in Table 2, its Walsh coefficients in Table 4, and its Walsh canonical representation in Table 6. Orders and defining lengths for partitions in this problem are noted in Table 7.

j	Partition (*J*)	Order	Defining Length	Walsh Coefficient (*w(j)*)
0	000	0	0	0
1	001	1	0	4
2	010	1	0	3
3	011	2	1	2
4	100	1	0	6
5	101	2	2	0
6	110	2	1	-10
7	111	3	2	0

Table 7

The largest order of a partition that has a non-zero Walsh coefficient indicates the maximum number of actions that influence each other's reward-gaining effects. Therefore, we will refer to the maximum order of partitions that have non-zero Walsh coefficients as the *effective action size* of the given RL task. In the case of Table 7, the effective action size is 2.

The maximum defining length of a partition that has a non-zero Walsh coefficient indicates the maximum number of time steps between actions that influence each other's reward-gaining effects. Therefore, we will refer to the maximum defining length of partitions that have non-zero Walsh coefficients as the *effective action span*. Note that the effective action span cannot be less than the effective action size minus one, by definition. In the case of Table 7, the effective action span is 1.

Note that these factors explain why one step look ahead was necessary and sufficient for obtaining the optimal reward value. One must consider actions of size 2, that are separated in time by no more than 1 step.

The effective action size and span are parameters that describe the complexity of a reinforcement learning task.

Moreover, these parameters are controllable through Walsh coefficient manipulation in problem design.

Pure Reward Delay

The taxonomy of complexity offered above leaves out one important aspect of RL tasks. To see this, consider the RL task depicted in Table 8, and its Walsh canonical representation, depicted in Table 9.

Time Step	Reward values *R*(*a*) (right = -1 action, left = +1 action)							
0	0							
1	5				-5			
2	-15		-5		5		15	
3	6	-6	6	-6	6	-6	-6	6

Table 8

Time Step	Reward values *R*(*a*) (right = -1 action, left = +1 action)							
0	0							
1	-5				5			
2	-5		5		-5		5	
3	6	-6	6	-6	6	-6	6	-6

Table 9

In the original representation of the task, some of the effects of the first action were delayed by one step. Therefore, a greedy approach will yield a suboptimal reward value of 6. However, in the Walsh canonical representation, this effect is removed. A greedy approach here yields the optimal reward value of 16. As this would suggest, examination of the Walsh coefficients (shown in Table 10) shows that the task has an effective action size of one and an effective action span of zero.

j	Partition (*J*)	Order	Defining Length	Walsh Coefficient (*w(j)*)
0	000	0	0	0
1	001	1	0	5
2	010	1	0	5
3	011	2	1	0
4	100	1	0	-6
5	101	2	2	0
6	110	2	1	0
7	111	3	2	0

Table 10

This shows that the key complexity of the task does not arise from either of these factors. This is a case of what we will refer to as *pure reward delay*.

Although the Walsh canonical representation removes pure reward delay, one can introduce arbitrary levels of pure reward delay from the Walsh representation by delaying the time when a Walsh coefficient is added to the reward., and by introducing constant rewards that are removed at later time steps. The number of steps that these effects are delayed is the amount of pure reward delay. For instance, consider the non-canonical Walsh representation in Table 11, which delays *w(1)* one step, adds a temporary reward of 5 for one step, and yields the single step of reward delay shown in Table 8.

t	*j*	Sign of *w(j) in r*eward values *R*(*a(t))*							
0	*0*	+							
1	*delayed*	+5				-5			
2	*1*	-		-		-		-	
	2	-		+		-		+	
	3	-		-		+		+	
		-5		-5		+5		+5	
3	*4*	-	+	-	+	-	+	-	+
	5	+	-	+	-	-	+	-	+
	6	+	-	-	+	+	-	-	+
	7	-	+	+	-	+	-	-	+

Table 11

Using Walsh Coefficients to Design RL Tasks

The advantage of the Walsh approach and taxonomy offered above is that it casts the complexities of RL tasks into a set of definite parameters (the Walsh coefficients) and a few related parameters that directly effect RL task complexity (the effective action size, span, and the amount of pure reward delay). At first, the adjustment of the *2L* Walsh coefficients to arrive at a given task complexity may seem as difficult as adjusting the reward matrix directly. However, if one focuses on the effective action size, span, and pure reward delay, task design becomes much simpler. To develop a RL task with a given level of complexity, one could set all Walsh coefficients that have the same order and the same defining length to be equal.

It is important to note that although action size, span and reward delay indicate the potential complexity of an RL task, they do not tell the entire story. For instance, consider a task that has an action size of three. In such a task it may be possible to obtain optimal performance by considering only an action span of one. This is possible because the relative value of immediate reward may override the effects of reward from any three coupled actions. Moreover, the best behavior for three separate actions can also give the best behavior for combinations of three actions.

However, one can design RL tasks where the effects of larger combinations of actions override those of smaller combinations (or vice versa) by considering the relative sizes of Walsh coefficients. A complete discussion of these effects is beyond the scope of this paper. A more thorough consideration of these issues can be gleaned from genetic algorithms analysis (Goldberg, 1989b).

Moreover, one can design RL tasks by randomly assigning values to appropriate Walsh coefficients, selected to insure desired values of action size, span, and reward delay. Then, by considering probability distributions of the relative sizes of Walsh coefficients, the expected complexity of a suite of such problems can be derived. Such problem design techniques are similar to those used in NK landscapes (Kauffman, 1993). A thorough consideration of the probabilities involved in this design technique is being developed (Smith and Smith, in preparation).

Extensions

The previous discussion employs undiscounted, binary, deterministic RL tasks. However, the techniques discussed can be easily extended to a much broader category of RL problems. Although, complete arguments and examples of these extensions cannot be included here for the sake of brevity, some discussion of these extensions is useful.

The reward measure used in the previous discussion was undiscounted. A more commonly used measure of performance is a discounted sum of rewards:

$$R(a) = \sum_{t=1}^{L} \gamma^t r(t, s(t), a(t))$$

where $0<\gamma<1$ is the *discount parameter*. Walsh coefficients can be derived from this measure of performance in the same manner discussed before. However, when designing a task using the Walsh canonical form, one must take discounting into account. This can be done by determining the coefficients as in the undiscounted case, and then multiplying the values of the coefficient combinations at each time step *t* by $(1/\gamma)$t.

To extend the techniques discussed here to decision tasks where more than two action alternatives are available at each state, one only need realize that a binary number can be assigned to actions at each state. By examining the form of the Walsh transform over a complete action sequence in this form, one can reformulate notions of effective action size, span, and pure reward delay.

The techniques can also be extended to non-deterministic tasks through appropriate consideration of the expected value of reward.

Final Comments

The suggested Walsh transform-based method offers several advantages for the design of RL test environments. Specifically:

- 〈 It provides for specific, tunable parameters to control problem complexity (i.e., the Walsh coefficients).
- 〈 It provides a set of separate parameters they accurately describe problem complexity:
  - ← The effective action size,
  - ← The effective action span, and
  - ← The amount of pure reward delay.
- 〈 It provides a framework for generating an infinite number of test problems of known complexity for testing RL methods.
- 〈 It allows RL test problem designers to draw on genetic algorithm analysis, where using Walsh transforms in problem design is well established. (Goldberg, 1989a; 1989b; 1989c; Smith and Smith, in preparation)

Given the suggested techniques, detailed, careful comparisons of RL methods can be performed. Given the key role such methods play in AI in intelligent control, such comparisons are of paramount importance

References

Barto, A. G., Sutton, R. S., and Watkins, C. J. C. H. (1989). Learning and Sequential Decision Making, COINS Technical Report, 89-95, University of Massachusetts, Amherst, MA

Barto, A. G. (1990). Some Learning Tasks From a Control Perspective, COINS Technical Report 90-122, University of Massachusetts, Amherst, MA

Beauchamp, K. G. (1984). *Applications of Walsh and Related Functions*. New York: Academic Press.

Goldberg, D. E. (1989a). Genetic algorithms and Walsh Functions: Part I, A Gentle Introduction. *Complex Systems* 3: 129-152.

Goldberg, D. E. (1989b). Genetic algorithms and Walsh Functions: Part II, Deception and Its Analysis. *Complex Systems* 3: 153-171.

Goldberg, D. E. (1989). *Genetic algorithms in search, optimization, and machine learning*. Reading, MA: Addison-Wesley

Holland, J. H., Holyoak, K. J., Nisbett, R. E., and Thagard, P. R. (1986). *Induction: Processes of Inference, Learning, and Discovery*. Cambridge, MA: MIT Press.

Kauffman, S. A. (1993). *The Origins of Order: Self-Organization and Selection in Evolution*. New York: Oxford University Press.

Smith, R. E. and Smith, J. (in preparation). Walsh Analysis of Random Landscapes. Intelligent Computing Systems Centre. The University of The West of England.

Watkins, C. J. C. H. and Dayan, P. (1992). Technical Note: Q-Learning. *Machine Learning* 8: 279-292.