Optimal Process Control of Symbolic Transfer Functions

C. Griffin and E. Paulson
Applied Research Laboratory
Penn State University
University Park, PA 16802
griffinch@ieee.org, ecp141@psu.edu

ABSTRACT

Transfer function modeling is a standard technique in classical Linear Time Invariant and Statistical Process Control. The work of Box and Jenkins was seminal in developing methods for identifying parameters associated with classical (r, s, k) transfer functions.

Computing systems are often fundamentally discrete and feedback control in these situations may require discrete event systems for modeling control structures and process flow. In these situations, a discrete transfer function in the form of an accurate hidden Markov model of input/output relations can be used to derive optimally responding controllers.

In this paper, we extend work begun by the authors in identifying symbolic transfer functions for discrete event dynamic systems (Griffin et al. Determining A Purely Symbolic Transfer Function from Symbol Streams: Theory and Algorithms. In *Proc. 2008 American Control Conference*, pgs. 1166-1171, Seattle, WA, June 11-13, 2008). We assume an underlying input/output system that is purely symbolic and stochastic. We show how to use algorithms for estimating a symbolic transfer function and then use a Markov Decision Processes representation to find an optimal symbolic control function for the symbolic system.

1. INTRODUCTION

Transfer function modeling is critical in minimum mean square error (MMSE) control [1]. The work of Box and Jenkins was seminal [2] and has been extended and enhanced over the years by several authors.

We contrast this with the discrete event control literature. In that case, plant models are often developed by hand. This may be reasonable in some cases but for real-world applications controllers need to be synthesized for complex (e.g., computational) systems that are not fully known a priori. In particular, it is difficult to be certain that manually created models accurately reflect plant dynamics. This is especially true when system transitions follow probability distributions. If we could automatically derive a plant model whose outputs are observed responses to a known set of inputs, the resulting model of input-output relationships would be a discrete event transfer function. This transfer function could be used to synthesize a discrete event controller that could optimize some objective function defined in terms of the discrete event dynamical system.

In [3], we showed how to extend Crutchfield and Shalizi's CSSR algorithm [4–6] to identify an asymptotically optimal Mealy Machine representation, when three parameters are supplied: l_1, the maximal input history length; l_2, the maximal output history length; and k the delay. We called this the *symbolic transfer function*. In this paper, we show how to use the derived system to optimize the return in a discounted reward context, essentially showing that this is equivalent to finding the solution to a Markov Decision Process [7]. It should be noted that this work is similar in spirit to the work of Watkins [8] and the Q-learning literature [9], which derive an optimal response to a Markov decision process when limited initial information on the reward structure is available. This work is different from the Q-learning literature in that (i) The Q-learning literature assumes an underlying Markov Decision Process (MDP) structure with only reward outputs; i.e., there is no input-output assumption. (ii) In Q-learning, reward is only a function of input, not input and output. (iii) There is no underlying assumption of lagged outputs as a function of inputs. (iv) Q-learning is a fundamentally online process that attempts to learn a system as it evolves. In this paper, we investigate a learning and optimization framework that operates offline. Thus the work is in perfect analogy to the Box-Jenkins control work [2].

2. PRELIMINARIES

In this section we provide the notation and preliminaries necessary for the proposed approach. Our notation is derived from Box and Jenkins [2] and symbolic dynamical systems [10] using time series expressed as a string of symbols. We discuss probabilistic automata, including probabilistic labeled transition systems (which are essentially Markov chains with labels) and probabilistic Mealey machines.

2.1 Time Series of Symbols

Let $\mathcal{A}$ (the *input* alphabet) and $\mathfrak{A}$ (the *output* alphabet) be finite sets of symbols. A symbolic time series is a sequence: $\mathbf{x} = \ldots x(-2)x(-1)x(0)x(1)x(2)\ldots$, where $x(t)$ represents the symbol that occurred at discrete time t in $\mathbf{x}$. If $x(t)$ is undefined, then we assume it is ϵ the empty symbol. Following the work of Box and Jenkins, let $\mathcal{B}$ be the backshift operator, so that $(\mathcal{B}\mathbf{x})(t) = x(t-1)$.

The set of all finite length strings of symbols from $\mathcal{A}$ and the concatenation $(+)$ operation form a monoid with unit ϵ. We may therefore write:

$$\mathbf{x} = \sum_{t=-\infty}^{\infty} x(t). \tag{1}$$

Consider a finite sub-sequence $t = 0$ to $t = s$ of $\mathbf{x}$, this may be written as:

$$(1 + \mathcal{B} + \dots \mathcal{B}^s)x(s) = A(\mathcal{B})x(s). \tag{2}$$

Let $A(\mathcal{B}) = \sum_{j=1}^{l} \mathcal{B}^j$.

We assume the following system dynamics

$$y(t) \sim \xi(\mathcal{B}^k A_1(\mathcal{B})x(t), A_2(\mathcal{B})y(t)) \tag{3}$$

where ξ is a function with output a probability distribution over $\mathfrak{A}$ and input recent observations (subsequences of $\mathbf{x}$ and $\mathbf{y}$). The observation $y(t)$ is drawn from the distribution provided by η. The formal polynomial A_1 has degree l_1 and the polynomial A_2 has degree l_2. Thus our models have three parameters (l_1, l_2, k). Without loss of generality, we can always assume $k = 1$ by appropriately shifting the output sequence $\mathbf{y}$ and adjusting l_1 and l_2 as needed.

If we assume an open loop control, then we have input dynamics:

$$x(t) \sim \eta\left(A_1(\mathcal{B})x(t)\right) \tag{4}$$

Closed loop control dynamics will be similar to those of Equation 3 but with outputs in $\mathcal{A}$.

The function ξ is a symbolic transfer function that maps inputs to outputs. If η and ξ are known, then a complete characterization of the (stochastic) system behavior is possible. The control problem, given ξ, is to determine an appropriate η satisfying certain objectives. If ξ is unknown, then the control problem is to first find a representation of ξ and then to determine a corresponding η.

In this paper we use the fixed length left shift operator. Suppose that $\mathbf{w} = a_1 a_2 \cdots a_n \in \mathcal{A}$. Let $\zeta(\mathbf{w}, a_{n+1}) = a_2 \cdots a_n a_{n+1}$ be the fixed length left shift operator.

2.2 Probabilistic Automata

A *labeled transition system* (LTS) is a tuple $G = \langle Q, \mathcal{A}, \delta \rangle$, where Q is a finite set of states, $\mathcal{A}$ is a finite alphabet and $\delta \subseteq Q \times \mathcal{A} \times Q$ is a transition relation. The transition relation δ is deterministic when for all $q \in Q$ and for all $x \in \mathcal{A}$ there is at most one $q' \in Q$ such that $(q, x, q') \in \delta$.

A *probabilistic LTS* is a pair $\langle G, p \rangle$ where G is an LTS and $p : \delta \to [0, 1]$ is a probability function such that

$$\sum_{a \in \mathcal{A}, q' \in Q} p(q, a, q') = 1 \quad \forall q \in Q. \tag{5}$$

It is easy to see that if there is some initial probability distribution π_0 over Q, then the triple $\langle G, p, \pi_0 \rangle$ is a Markov chain with transitions labels.

A *polygenic Mealy machine* is a tuple $M = \langle Q, \mathcal{A}, \mathfrak{A}, \delta \rangle$ where Q and $\mathcal{A}$ are as above and $\mathfrak{A}$ is a second *output* alphabet and $\delta \subseteq Q \times \mathcal{A} \times \mathfrak{A} \times Q$ is a transition relation with input alphabet $\mathcal{A}$ and output alphabet $\mathfrak{A}$. Determinism of the transition relation is defined just as it was for an LTS. If we assign a probability function to the transition relation (as we did for probabilistic LTS), then we obtain a probabilistic Mealy machine. (Note for the sake of brevity, we remove the term polygenic.)

2.3 Probability Distribution Functions of Symbols and Symbolic Transfer Functions

Consider an open loop control function $\eta : \mathcal{A}^{l_1} \to \mathcal{F}_{\mathcal{A}}$, where $\mathcal{F}_{\mathcal{A}}$ is the set of probability distributions with support $\mathcal{A}$. The function η describes a probabilistic labeled transition system whose states are composed of strings of length l_1 from $\mathcal{A}$ and whose transitions are defined so that if $w_1, w_2 \in \mathcal{A}^{l_1}$ and $w_2 = \zeta(w_1, a)$ for $a \in \mathcal{A}$, then $p(w_1, a, w_2) = \eta(w_1)(a)$. It is worth noting that this *may* not be the smallest statistically equivalent probabilistic labeled transition system describing the behavior of η, however it is sufficient as a model of η.

The function $\xi : \mathcal{A}^{l_1} \times \mathfrak{A}^{l_2} \to \mathcal{F}_{\mathfrak{A}}$, describes a *formal* probabilistic Mealey machine whose states are composed of pairs of strings in $\mathcal{A}^{l_1} \times \mathfrak{A}^{l_2}$ and whose transitions are defined so that if $(w_1, v_1), (w_2, v_2) \in \mathcal{A}^{l_1} \times \mathfrak{A}^{l_2}$ and $w_2 = \zeta(w_1, a)$ and $v_2 = \zeta(v_1, \alpha)$, then:

$$p\left[(w_1, v_1), a, \alpha, (w_2, v_2)\right] = \xi(w_1, v_1)(\alpha) p_a$$

where p_a is a *variable* holding the probability that a occurs.

3. EXTRACTING ξ FROM DATA

A method for deriving η and ξ from data (as state machines) is provided in [3]. We review the results here and simplify the work presented in [3]. We focus specifically on generating ξ, since the algorithms for deriving η are discussed extensively in [3, 4, 6]. We simplify the discussion in [3].

Assume we are given $\mathbf{x}$, the input and $\mathbf{y}$, the output. We will assume that the input signal is generated randomly and not as a function of the observed output (i.e., $\mathbf{x}$ is dithered). If not, we will be modeling the coupled dynamics of an existing control system and we will not obtain a true representation of the impact of a control signal on system output. This is consistent with classical transfer function modeling.

Let $\mathbf{w}$ be a subsequence of $\mathbf{x}$ and let $\mathbf{z}$ be a subsequence of $\mathbf{y}$. By $(\mathbf{w}, \mathbf{z})$ we mean the pair of input/output sequences. Assume that we know there is a lag of k time observation symbols before an output is observed. Then, for example if we began generating input symbols $x(1)x(2)\cdots$, and the lag is 2, then the output $y(1)$ will occur as symbol $x(3)$ is generated. From now on, we assume that the two sequences are appropriately aligned, so that $y(2)$ corresponds to the input $x(1)$, and previous output $y(1)$ even though though $y(2)$ appears at time $k + 1$.

Algorithm 1 will produce the symbolic transfer function; it is similar to the initialization portion of the CSSR algorithm presented by Crutchfield and Shalizi [4–6]. Let $\#(\mathbf{w}_1, \mathbf{x}, \mathbf{w}_2, \mathbf{y})$ be the number of times sequence $\mathbf{w}_1$ occurs in $\mathbf{x}$ and at the same time sequence $\mathbf{w_2}$ occurs in $\mathbf{y}$ so that the ends of the two sequences coincide. (Recall, $\mathbf{w}_1$ may have different length $\mathbf{w}_2$). Also let $\#(\mathbf{w}_1, \mathbf{x}, \mathbf{w}_2, \mathbf{y}, \alpha)$ be the number of times $\mathbf{w}_1$ occurs in $\mathbf{x}$ and at the same time sequence $\mathbf{w_2}$ occurs in $\mathbf{y}$ so that the ends of the two sequences coincide and α immediately follows $\mathbf{w}_2$.

Using the remarks presented in Section 2.3, it is easy to see how to construct a stochastic Mealey machine from the resulting ξ function. In Line 1 of Algorithm 1 we compute

Algorithm 1

```
Input:
  Symbolic time series $\mathbf{x}$ and $\mathbf{y}$
  Alphabets $\mathcal{A}$ and $\mathfrak{A}$
  Lengths $l_1$ and $l_2$
Output:
  Map $\xi$
Procedure:
 1: Let $W$ be the set of pairs of substrings of $(\mathbf{w}_1, \mathbf{w}_2)$ with length
    $l_1$ and $l_2$ respectively, synchronized in ending position.
 2: for all $(\mathbf{w}_1, \mathbf{w}_2) \in W$ do
 3:     Compute $\Pr(\alpha \in \mathfrak{A}|\mathbf{w}_1, \mathbf{w}_2) := \frac{\#(\mathbf{w}_1,\mathbf{x},\mathbf{w}_2,\mathbf{y},\alpha)}{\#(\mathbf{w}_1,\mathbf{x},\mathbf{w}_2,\mathbf{y})}$.
 4: Define $f_{\mathbf{w}_1,\mathbf{w}_2}(\alpha) := \Pr(\alpha \in \mathfrak{A}|\mathbf{w}_1, \mathbf{w}_2)$.
 5: Define $\xi(w_1, w_2) := f_{\mathbf{w}_1,\mathbf{w}_2}$.
 6: return $\xi$
```

the substrings needed. In Line 3, we produce the maximum likelihood estimator for the probability of observing output a given observation of input sequence $\mathbf{w}_1$ and output sequence $\mathbf{w}_2$ prior to the appearance of a. Finally, ξ is defined as the collection of these estimators.

3.1 Reducing the State Space

Assuming the dynamics provided in Equation 3, Algorithm 1 produces a state space $Q = \mathcal{A}^{l_1} \times \mathfrak{A}^{l_2}$, where states correspond to pairs of histories of length l_1 and l_2. At state $q = (\mathbf{w}_1, \mathbf{w}_2) \in Q$ input $a \in \mathcal{A}$ occurs with corresponding output $\alpha \in \mathfrak{A}$ and transfers the system to a new state $q' = (\mathbf{w}_1', \mathbf{w}_2') = (\zeta(\mathbf{w}_1, a), \zeta(\mathbf{w}_2, \alpha))$. We will write $q' = \delta(q, a, \alpha)$ to denote this relationship.

For any state $q \in Q$, $\xi(q) \in \mathcal{F}_{\mathfrak{A}}$. The probability distributions can be thought of as a set of vectors in $[0, 1]^{|\mathfrak{A}|}$, for a fixed ordering of $\mathfrak{A}$. For the remainder of this section, we will treat $\xi(q)$ as such a vector.

In [3], Algorithm A provides a way of reducing this state space that can be useful for simplifying the optimization problem presented in the next section. The technique we present is identical to the one given in [3] and is drawn from the results of Shalizi and Crutchfield [4–6]. We present it in a manner more consistent with modern pattern recognition techniques.

The approach to minimizing the state space Q is to first cluster (merge) existing states that have statistically identical conditional distributions over $\mathfrak{A}$ and then to de-cluster those states that would result in a non-deterministic state transition function δ.

Let $q_1, q_2 \in Q$. Define a metric $T(q_1, q_2)$. In [3] the metric was based on the p-value of the Kolmogorov-Smirnov test when applied to distributions $\xi(q_1)$ and $\xi(q_2)$. A χ^2 test [11], multinomial comparison test [12], or simple Euclidean distance could also be used. Let t_0 be a user provided threshold and let $p = \{q_{i_1}, \ldots, q_{i_n}\}$ be a cluster of states in Q. Define

$$\xi(p) = \frac{1}{n}\sum_{j=1}^{n} \xi(q_{i_j}) \qquad (6)$$

We can then define $T(q, p)$ using $\xi(p)$ as expected. Clustering the states of Q into a new state space P can now proceed iteratively. Algorithm 2 shows this procedure.

Clearly the size of state space P is less than or equal to the size of state space Q. However, the resulting transition function δ may be non-deterministic.

To correct the non-determinism, we must split the states

Algorithm 2

```
Input:
  State Space $Q$
  Threshold $t_0 \in \mathbb{R}$
Output:
  State Space $P$
Procedure
 1: $P := \emptyset$
 2: for all $q \in Q$ do
 3:   if $P \neq \emptyset$ then
 4:     $p^* = \arg\min_{p \in P} T(q, p)$
 5:     if $T(q, p^*) > t_0$ then
 6:       $p_{\text{new}} = \{q\}$
 7:       $P := P \cup \{p_{\text{new}}\}$
 8:     else
 9:       $p := p \cup \{q\}$
10:   else
11:     $p_{\text{new}} = \{q\}$
12:     $P := P \cup \{p_{\text{new}}\}$
```

of P apart into the final (reduced) state space R. Splitting is a recursive operation in which we iterate through the states of P selecting an initial pair $(\mathbf{w}_1, \mathbf{w}_2) = q$ in some state p. We create a new state $r \in R$ with this q. We then analyze the remaining elements (original states) in p to determine whether they should be added to r or whether a new state r' should be created in R to deal with non-determinism in the transition function. This procedure is repeated recursively until no new states are added to R. Algorithm 3 summarizes the procedure.

Algorithm 3

```
Input:
  State Space $P$
Output:
  State Space $R$
Procedure
 1: $N_0 := 0$
 2: $N := |P|$
 3: repeat
 4:   for all $p_i \in P$ do
 5:     $s := 0$
 6:     $M = |p_i|$ {The size of cluster $p_i$}
 7:     Choose $q_{i_0} \in p_i$
 8:     Create state $r_{is} = \{q_{i_0}\}$
 9:     $R := R \cup \{r_{is}\}$
10:     for all $q_{ij} \in p_i$ $(j \neq 0)$ do
11:       FOUND $= 0$
12:       for $l \in 0, \ldots, s$ do
13:         Choose $q \in r_{il}$
14:         if $\delta(q, a, \alpha) = \delta(q_{i_j}, a, \alpha)$ for all $(a, \alpha) \in \mathcal{A} \times \mathfrak{A}$
            then
15:           FOUND $= 1$
16:           $r_{il} := r_{il} \cup \{q_{i_j}\}$
17:       if FOUND $= 0$ then
18:         $s := s + 1$
19:         Create state $r_{is} = \{q_{i_j}\}$
20:         $R := R \cup \{r_{is}\}$
21:   $N_0 := N$
22:   $N : |R|$
23: until $N \neq N_0$
```

Beginning with an initial symbolic input/output pair $(\mathbf{x}, \mathbf{y})$ the application of Algorithms 1-3 will create a reduced state space representation of the symbolic transfer function ξ. It should be noted that it is sufficient to execute only Algorithm 1 to obtain a complete representation of the symbolic transfer function. If the state space is too large, then the additional algorithms can be executed to enhance processing later.

Work in [3, 5, 6] is sufficient to show that the resulting ξ (and by extension η function) representations are minimal in state size after executing Algorithms 1 - 3 as the number of samples approaches infinity. More recent work by Paulson and Griffin shows how to correct this for smaller sample sizes [13] and also shows that the minimum state estimation problem is NP-hard in this case. State minimization is useful in managing the size of the optimal control problem as we see in the sequel.

4. OPTIMAL CONTROL OF THE SYMBOLIC TRANSFER FUNCTION

In this section, we assume known and fixed alphabets $\mathcal{A}$ and $\mathfrak{A}$. We further assume that l_1, l_2 and k are known and fixed. An algorithm for inferring l_1 is available in [14]. It can be extended to infer l_2 if needed. As noted previously, we will assume that $k = 1$, which will simplify the notation significantly.

Let Q be the state space of ξ and let $r_q : \mathcal{A} \times \mathfrak{A} \to \mathbb{R}$ be a state parameterized reward function. At each discrete time t the system is in some state $q \in Q$. A control function chooses an input $a \in \mathcal{A}$ which causes an output symbol α to be generated at the next time step. This output is a function of previous inputs up to (but not necessarily including a, when there is a $k = 1$ lag). We may assume a reward $\beta^t r_{q(t)}$ results where $\beta \in (0, 1)$ is a discounting factor. The state then becomes q' as a result of the input and output.

For all $q \in Q$ define:

$$R_q = \begin{bmatrix} r_q(a_1, \alpha_1) & \cdots & r_q(a_1, \alpha_n) \\ \vdots & \ddots & \vdots \\ r_q(a_m, \alpha_1) & \cdots & r_q(a_m, \alpha_n) \end{bmatrix} \tag{7}$$

where $|\mathcal{A}| = m$ and $|\mathfrak{A}| = n$. For state $(\mathbf{u}, \mathbf{v}) \in q \in Q$ let $\xi(q) = \xi(\mathbf{u}, \mathbf{v})$ be the vector of probabilities that the various $\alpha \in \mathfrak{A}$ will occur.

If we are given the symbolic transfer function ξ and the objective is to derive a policy η so that the long run pay-off is optimized under the β-discounting rule, then this problem can be coded as a Markov Decision Problem [7]. Following [15], we may write the problem of maximizing long run reward subject to β discounting as:

$$\begin{cases} \min & \displaystyle\sum_{q' \in Q} \pi^0_{q'} v_{q'} \\ \text{s.t.} & v_q \geq [R_q \xi(q)]_a + \beta \displaystyle\sum_{q' \in Q} \Pr(q'|q, a) v_{q'} \quad \forall q \in Q, a \in \mathcal{A} \end{cases} \tag{8}$$

Here $\mathbf{v}$ is a vector in $\mathbb{R}^{|Q|}$ with elements v_q and for vector y, $[y]_a$ indicates the element of y corresponding to $a \in \mathcal{A}$. Note well: the problem structure itself (from ξ) is incorporating the delay in output response k. This allows us to use the MDP framework for the more general problem. Problem 8 has known dual:

$$\begin{cases} \max & \displaystyle\sum_{q \in Q} \sum_{a \in \mathcal{A}} [R_q \xi(q)]_a x_{qa} \\ \text{s.t.} & \displaystyle\sum_{q \in Q} \sum_{a \in \mathcal{A}} \left[\delta(q, q') - \beta \Pr(q'|q, a)\right] x_{qa} = \pi^0_{q'} \quad \forall q' \in Q \\ & x_{qa} \geq 0 \quad \forall q \in Q, a \in \mathcal{A} \end{cases} \tag{9}$$

Here x_{qa} are the dual variables corresponding to the $|Q| \times |\mathcal{A}|$ constraints in Problem 8 and $\delta(q, q')$ is the Dirac delta function. It should also be noted that $Pr(q'|q, a)$ is precisely $\xi(q)(\alpha)$ where $q' = \delta(q, a, \alpha)$.

Proposition 1. *Let $\mathbf{x}^*$ be an optimal (vector) solution to Problem 9. For fixed $q \in Q$, let*

$$x_q = \sum_{a \in \mathcal{A}} x_{qa} \tag{10}$$

Then the optimal policy η is given by:

$$\eta(q)(a) = \frac{x^*_{qa}}{x^*_q} \tag{11}$$

For fixed $\mathbf{u} \in \mathcal{A}^{l_1}$, if for all $\mathbf{v}_1, \mathbf{v}_2 \in \mathfrak{A}^{l_2}$ we have $\eta(\mathbf{u}, \mathbf{v}_1) = \eta(\mathbf{u}, \mathbf{v}_2)$ then η is an open loop controller. Note Problem 9 has a constraint space with size equal to the number of states in Q, making state minimization of the η model useful for minimizing the size of the optimization problem. Even without state minimization, recent work by Post and Ye [**?**] has shown that the simplex algorithm is strongly polynomial for Markov decision problems, like the one formulated. Thus scaling will by polynomial.

5. SYMBOLIC STATISTICAL PROCESS CONTROL

We define a learning and optimization strategy for symbolic dynamical systems analogous to the statistical process control process identified by Box and Jenkins and discussed in detail in [1].

Assuming an N-time step learning period, the following algorithm can be used to derive a symbolic transfer function and optimal open-loop η controller:

1. While $t \leq N$
   (a) Choose a completely random (i.e., dithered) input (i.e., a random η function) and record the output.
   (b) Builds symbolic sequences $\mathbf{x}$ and $\mathbf{y}$.
2. At time $t = N$, use Algorithms 1 - 3 (or Algorithm A from [3] to derive function ξ, the symbolic transfer function.
3. Using function ξ, compute η^* using Problem 9.

5.1 Example: Learning to Play against Tit-for-2-Tats

Repeated games offer a simple symbolic input / output system on which to build an example. This modeling approach could also be used in other computational systems in which symbolic (categorical) inputs lead to symbolic (categorical) outputs like in I/O protocols.

The Prisoner's Dilemma Game is a well known symmetric bimatrix game describing the behavior of two captured criminals. There are two strategies for each player, Collude (C)

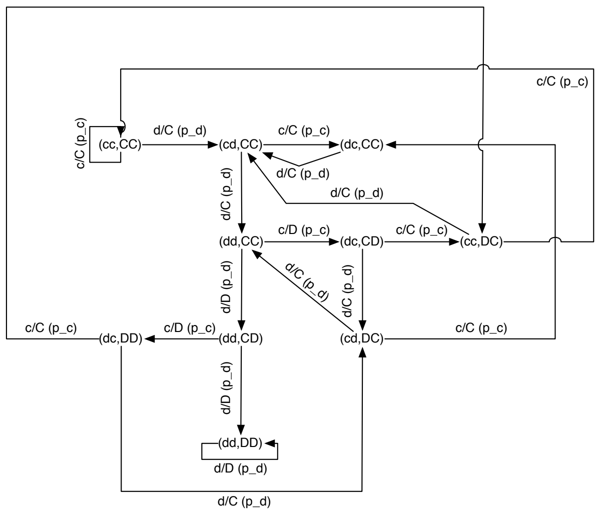

Figure 1: The formal Mealey machine state space model of ξ derived using Algorithm 1 from a Tit-for-Two-Tats game scenario. In this figure, the symbols p_c and p_d indicate the probability that Player 1 strategies c or d will be played. The deterministic nature of the Tit-for-Two-Tats algorithm makes it deterministic in Player 2's strategy.

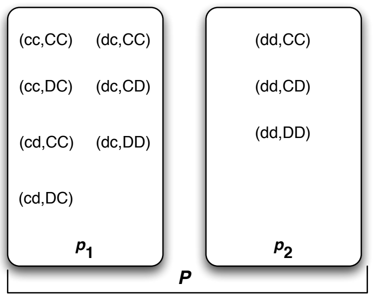

Figure 2: The clustering of states from the formal Mealey state machine shown in Figure 1.

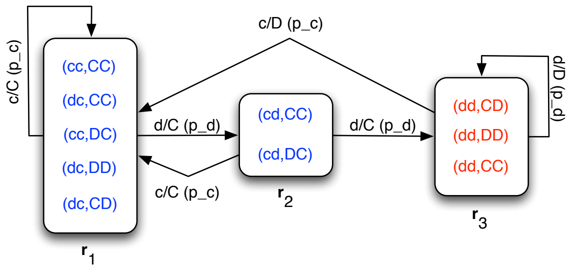

Figure 3: The reduced state space formal Mealey machine modeling ξ derived from an input sequence of player moves. This machine is the result of executing Algorithms 1 - 3.

and Defect (D). ([16], Page 25) has an excellent overview. In the repeated Prisoner's Dilemma Game, two players repeat this game in an attempt to maximize their long run pay-off.

In Tit-for-2-Tats, a forgiving strategy is used to avoid unending defection. If Player 1 plays C in round i, then Player 2 will play C in round $i+1$. If Player 1 plays D in round i and had previously played C in round $i-1$, then Player 2 still plays C in Round $i+1$. Otherwise, Player 2 plays D. If Player 2 is using a fixed strategy (protocol) $\mathcal{S}$ such as Tit-for-2-Tats, it may be possible to *game* the strategy thus deriving a better long run payoff for Player 1. This is exactly the problem of generating an optimal control law for Player 1.

5.1.1 *Learning ξ from Input Data*

We apply Algorithms 1 - 3 to identify a ξ function for the Tit-for-2-Tats strategy assuming Player 1 executed a random strategy in order to fully explore the search space. As in [3], we used 25 iterations of a repeated game to extrapolate ξ using a randomized input.

After executing Algorithm 1 on the input sequence, we obtain the state space model for the behavior of ξ shown in Figure 1. This model shown in Figure 1 contains 10 states, corresponding to the 10 observed pairs of strategy sequences observed for Players 1 and 2, each with length 2. We can cluster theses states using Algorithm 2. The results of running Algorithm 2 are shown in Figure 2. Having identified the clustered state space, we can execute Algorithm 3 to identify the reduced state space and formal Mealey machine modeling ξ. This is shown in Figure 3.

5.1.2 *Finding the Optimal Response η*

Let:

$$R_q = \begin{bmatrix} 1 & -2 \\ 2 & \frac{1}{2} \end{bmatrix}$$

for all $q \in Q$. This is a standard Prisoner's Dilemma type game matrix. We suppose that the strategies are read from left to right as collude or defect and from top to bottom in the same order. If we assume an initial state to contain $([c,c],[C,C])$, i.e., one in which all players have cooperated up to this point, then when using the derived ξ function, the specific instance of Problem 9 is:

$$\begin{cases} \max\ x_{1,c} + 2x_{1,d} + x_{2,c} + 2x_{2,d} - 2x_{3,c} + \dfrac{1}{2}x_{3,d} & \\ s.t.\ \ (1-\beta)x_{1,c} + x_{1,d} - \beta x_{2,c} - \beta x_{3,c} = 1 & (q'=1) \\ \quad -\beta x_{1,d} + x_{2,c} + x_{2,d} = 0 & (q'=2) \\ \quad (1-\beta)x_{3,d} - \beta x_{2,d} + x_{3,c} = 0 & (q'=3) \\ \quad x_{1,c}, x_{2,c}, x_{3,c}, x1, d, x_{2,d}, x_{3,d} \geq 0 & \end{cases}$$

The state q' to which each constraint corresponds is shown in the right most column. For large enough (e.g., $\beta > 1/2$) the optimal solution yields behavior:

$$\eta([c,c])(d) = 1.0$$
$$\eta([c,d])(c) = 1.0$$
$$\eta([d,c])(d) = 1.0$$

and $\eta(q)(a) = 0$ for all other $q \in Q$ and $a \in \mathcal{A}$. These dynamics create a cycle in which Player 1 (who constructs η) will repeatedly defect and then collude, thus taking advantage of the altruism of Player 2 in employing the Tit-for-2-Tats strategy. This is illustrated in Figure 4. It is worth

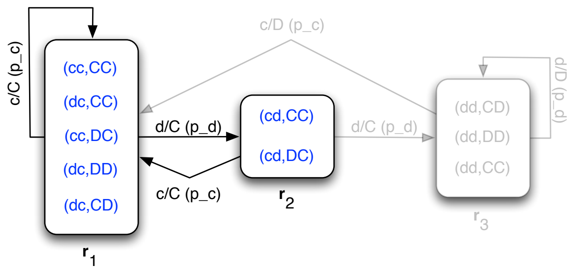

Figure 4: The cycle induced by the dynamics provided in the optimal response to Tit-for-Two-Tats illustrated as a sub-machine of the reduced state space shown in Figure 3.

noting that for smaller values of β (i.e., $\beta < 0.45$) the optimal strategy is to always defect. Thus when the award decay is too great, cyclic cooperation and defection is not worth it.

6. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we have provided a method for optimizing a control process in which the inputs and output are symbolic and the transfer function is stochastic. This work attempts to apply the notions of optimal control in the Box Jenkins framework [2] to the case of purely symbolic processes. We drew upon our initial work in [3] to obtain symbolic transfer function estimators. We then applied results from Markov Decision Processes [7] to determine closed loop controls in this case.

For future work, we note that in computing an estimator for ξ, we applied Algorithms 1 - 3 (Algorithm A of [3]). These algorithms provides a pointwise predictor for the multinomial distributions that comprise the symbolic transfer functions. We showed in [17] how to use confidence intervals to improve recognition in hidden Markov models [18]. In that work, we showed how to derive confidence intervals on the transitions of a specific HMM process. The same technique can be applied here to derive confidence intervals on the transition probabilities of the symbolic transfer function. We could use this information to find a solution that is robust to our imprecise knowledge of the second player using a competitive Markov decision process [15] in this case as well. This is particularly useful for online learning when the convergence of the symbolic transfer function is slow.

Acknowledgement

Portions of Dr. Griffin's and Ms. Paulson's work were supported by the Army Research Office under Grant W911NF-11-1-0487.

7. REFERENCES

[1] E. del Castillo, *Statistical Process Adjustment for Quality Control.* Wiley Interscience, 2002.

[2] G. E. P. Box and G. M. Jenkins, *Time Series Analysis: Forecasting and Control*, 2nd ed. Holden-Day, 1976.

[3] C. Griffin, R. R. Brooks, and J. Schwier, "Determining a purely symbolic transfer function from symbol streams: Theory and algorithms," in *Proc. American Control Conference*, Seattle, WA, June 11-13 2008, pp. 4065–4067.

[4] C. Shalizi and J. Crutchfield, "Compuational mechanics: Pattern and prediction, structure and simplicity," *J. Statistical Physics*, vol. 104, no. 3/4, 2001.

[5] C. R. Shalizi, K. L. Shalizi, and J. P. Crutchfield, "Pattern discovery in time series, part i: Theory, algorithm, analysis, and convergence," Santa Fe Institute, Tech. Rep., 2002.

[6] C. Shalizi, K. Shalizi, and J. Crutchfield, "An algorithm for pattern discovery in time series," arXiv:cs.LG/0210025 v3, November 2002.

[7] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* New York, NY: John Wiley and Sons, 1994.

[8] C. Watkins, "Learning from Delayed Rewards," Ph.D. dissertation, Cambridge University, 1989.

[9] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman, "Pac model-free reinforcement learning," in *Proc. 23rd ACM International Confernce on Machine Learning International Confernce on Machine Learning*, 2006.

[10] M. Morse and G. Hedlund, "Symbolic dynamics," *American Journal of Mathematics*, vol. 60, no. 4, pp. 815–866, 1938.

[11] R. Hogg and E. Tanis, *Probability and Statistical Inference*, 7th ed. Pearson/Prentice-Hall, 2006.

[12] C. P. Quesenberry and D. C. Hurst, "Large sample simultaneous confidence intervals for multinomial proportions," *Technometrics*, vol. 6, pp. 191–195, 1964.

[13] E. Paulson and C. Griffin, "Computational Complexity of the Minimum State Probabilistic Finite State Learning Problem on Finite Data Sets," *ArXiv e-prints*, Dec. 2015.

[14] R. R. Brooks, J. Schwier, C. Griffin, and S. Bukkapatnam, "Zero knowledge hidden markov model inference," *Pattern Recognition Letters*, vol. 30, pp. 1273–1280, 2009.

[15] J. Filar and K. Vrieze, *Competitive Markov Decision Processes.* New York, NY, USA: Springer-Verlag, 1997.

[16] P. Morris, *Introduction to Game Theory.* Springer, 1994.

[17] R. R. Brooks, J. Schwier, and C. Griffin, "Behavior detection using confidence intervals of hidden markov models," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, vol. 39, no. 6, pp. 1484–1492, December 2009.

[18] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, 1989.