Consistent Measures and Bounds for Control Policies

Technical Report of the ISIS Group
at the University of Notre Dame
ISIS-93-001
January, 1993

Peter T. Szymanski and Michael D. Lemmon
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556
(219)-631-5480
e-mail: pszymans@maddog.ece.nd.edu

Interdisciplinary Studies of Intelligent Systems

Consistent Measures and Bounds For Control Policies

Peter T. Szymanski and Michael Lemmon *
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556
(219)-631-5480

February 3, 1993

Abstract

Many procedures exist for finding solutions to optimal control problems. Measures characterizing each method describe the method's performance as functions of quantities that the method utilizes. However, the measures are not usually consistent across solutions so as to allow unbiased comparison of different solutions to the same problem. This paper presents state entropy, control entropy, and average work — measures inspired by Source Coding Theory — that characterize the complexity of solutions, the solution environments, and the performance of a solution, all in a consistent fashion. It demonstrates that there is a trade-off between the complexity of a solution and its performance. It further presents a lower bound on the measures (Shannon's Bound) and presents Blahut's Algorithm that computes the bound. Finally, it demonstrates that Blahut's Algorithm may be useful in computing optimal policies subject to system constraints.

1 Introduction

Numerous procedures exist for deriving policies for optimal control problems. Heuristic[6, 13] and dynamic programming[2, 8] methods use either a priori knowledge of the workspace or prior experience to derive and execute optimal control policies. However, the concern of most of these methods is with the minimization of some cost functional. The methods generally

*The authors would like to acknowledge the partial financial support of the National Science Foundation (IRI91-09298)

do not address the complexity of the solution, nor do they provide any consistent measures to facilitate comparison with other methods.

Saridis and Valavanis[9, 14, 15] do address the area of solution complexity in intelligent control systems. They use information theoretic concepts of entropy and transmission of information (also known as average mutual information) to provide a framework for the discussion of intelligent systems. The framework quantifies the uncertainty or complexity of solutions and uses entropy to quantify system performance. Through minimization of the entropy, they determine the optimal system performance. However, their use of entropy does not directly incorporate any theoretical limits on performance as a function of the constraints of a system, nor does it provide any direct method for trading solution optimality for reduced solution complexity.

This work examines the use of source coding theory, a sub-field of information theory, to more thoroughly deal with solution performance and complexity. Source coding theory[5] extends Saridis' basic framework. It examines how the complexity of solutions can vary when system performance can be sub-optimal. That is, it studies the trade-off between solution complexity and system performance, where increasing one tends to decrease the other. It offers the Shannon Bound as a lower limit on system performance and solution complexity. Blahut's Algorithm[3] offers a computational method for computing the Shannon Bound and possibly for computing optimal control policies. Finally, the study results in a simple control network that uses the results of Blahut's Algorithm to control a system.

The remainder of this paper presents the details of the approach. Section 2 introduces notation and definitions. Section 3 presents entropy and work measures that quantify both space and time requirements. Section 4 presents bounds on the entropies of a solution and a bound on solution complexity as a function of the work involved. Section 5 demonstrates the concepts in a simple navigation problem. Finally, a discussion of the concepts and future work ends the paper in Section 6.

2 Preliminaries

Consider the setup of Figure 1 which presents two interconnected systems. The figure depicts the relationship between a controlling system and the workspace in which it operates. This configuration is very similar that of a controller (controlling system) and its plant (workspace) in control theory. It is also similar to the hybrid control system of [12], if the workspace is the discrete event system plant. The overall system operates through the interaction of the controlling system with the workspace via Γ and Λ. An illustration of such an interpretation of a system is a robot (controlling system) that senses and acts (via Λ and Γ) in a world (workspace) to accomplish some goal (control problem).

The mathematical model for the two systems is a pair of situated automata. Situated automata are a pair of automata with the output of one being the input to the other and vice versa. Let G be the workspace, and let $G = (X, U, X_F, f(x,u))$. G is a finite state automaton (FA). X is a finite set of states and $X \subset \mathbb{R}^n$. U is a finite set of controls and $U \subset \mathbb{R}^p$. X_F is a set of final or goal states, $X_F \subset X$. $f(x,u)$ is the state transition function such that $f : X \times U \to X$. G is such that $f(x,u)$ is deterministic (for all $x \in X$ if $u_1 = u_2$, then $f(x,u_1) = f(x,u_2)$) and invertible (for all u_k and $i, j \in X$ such that $f(i,u_k) \to j$, $\exists\ u_l = {u_k}^{-1}$ such that $f(j,u_l) \to i$). Determinism and invertibility taken together imply that the underlying directed graph for G is a *symmetric* graph. Finally, each control in the workspace has an associated cost for transitioning between the states. $c(x,u)$ represents the cost of executing u when in x, where $c(x,u) \in \mathbb{R}$ and $c(x,u) \geq 0$.

The model for the controlling system is another automaton. (Actually, it is just a very simple mapping from an input set to an output set, an automaton implementation is only required for modeling of the world internally by the controlling system.) Let $R = (\mathcal{P}, M, \mathcal{P}_F, \delta)$ be the mathematical model of the system. $\mathcal{P}$ is the set of perceptions or *percepts* that the system receives from its sensors with $\mathcal{P} \subset \mathbb{R}^m$. $M \subset \mathbb{R}^q$ is the set of programs that the controller issues to effect changes in the workspace. $\mathcal{P}_F$ is the set of the controller's perception of its goal states. There is a one-to-one mapping between $\mathcal{P}_F$ and X_F. The controller uses a control policy, δ, to map inputs to the appropriate outputs ($\delta : \mathcal{P} \to M$).

The last two features of Figure 1 are Λ and Γ. Λ maps the states of the workspace into the perceptions of the controlling system ($\Lambda : X \to \mathcal{P}$), while Γ maps the output programs of the controlling system to the controls of the workspace ($\Gamma : M \to U$). For the purposes of this paper, these two mappings are assumed to be one-to-one.

The following example illustrates the situated automata by detailing how a mobile robot might navigate a workspace. Consider a mobile robot which finds itself, at time instant n, in some state, $x^{(n)} \in X$, in the workspace (here the workspace includes both the robot and its environment). The robot's control system, R, determines its perception of the current state by computing $p = \Lambda(x^{(n)})$, where $p \in \mathcal{P}$. R uses p in conjunction with δ to determine which motor program to execute. R issues program $\alpha = \delta(p)$, which gets mapped to some u via Γ. The workspace (robot and environment) transitions to a new state by executing the issued control where $x^{(n+1)} = f(x^(n), u)$. The robot identifies its new state and repeats the procedure. Repetition of this sensing/planning/actuating procedure continues until the system perceives itself to be in a final state $x^{(f)} \in X_F$. At this point, navigation is complete as the robot encounters the goal.

3 Performance Measures

For a given workspace, various control policies perform differently. Performance measures usually characterize each policy. Current measures, such as those in Thrun[13], are typically complexity theoretic measures[7] that quantify space or time requirements. They indicate the worst case requirements for a control policy, and are usually specific to the implementation of the control algorithms. They are not necessarily consistent across algorithms because the measures usually depend upon how an algorithm represents the quantities.

This lack of independence of measures from their representations suggests the need for measures that are in some way standardized. A standard means for characterizing any control policy's resources is necessary. Such a characterization would lead to measures that quantify resources of different policies consistently. Consistent measures of space and time facilitate the comparison of policies and the derivation of absolute bounds on the measures. This section presents measures that are consistent across policies and characterize memory usage and policy performance. The first two quantify memory usage while the third quantifies performance.

Consider a set, V, whose elements, v, have an associated probability distribution, $p(v)$. Define the *entropy* of V as

$$H(V) = -\sum_{i=1}^{|V|} p(v_i) \log p(v_i) \tag{1}$$

where $|V|$ denotes the cardinality of the set. Entropy is an information theoretic measure that quantifies the uncertainty associated with a set. It also measures the self information of a set. If the interpretation of the log of the individual probabilities is as the amount of information that an element imparts by its occurrence, then the entropy is the average self information of the set.

Entropies characterize the states and actions in both the controlling system and the workspace. However, since Λ and Γ are one-to-one mappings, one need only consider the entropies of either the workspace or the system. The remainder of the paper concentrates on the entropies of the workspace.

The workspace consists of a set of states, X. Each state, $x \in X$ occurs with a frequency or probability, $p(x)$. The entropy of the ensemble of states, the *state entropy*, is

$$H(X) = -\sum_{i=1}^{|X|} p(x_i) \log p(x_i) \tag{2}$$

Again, since $\mathcal{P}$ is equivalent to X ($p(x) = p(p)$ for all $x \in X$ and $p \in \mathcal{P}$ such that $\Lambda(x) \rightarrow p$), the entropy of the percepts equals the state entropy ($H(\mathcal{P}) = H(X)$).

State entropy characterizes the memory requirements for the workspace or system as follows. State entropy, and any entropy as defined in (1), has an upper bound. The bound is a result of the divergence inequality and is

$$H(X) = -\sum_{i=1}^{|X|} p(x_i) \log p(x_i) \leq \log |X| \tag{3}$$

The equality holds if the $p(x)$'s are uniformly distributed. If the probability distribution tends to be uniform, it means that the states in the ensemble occur with the same or similar probabilities. This means that a system will see a greater variety of states on the average, and it will have to be able to store or identify the controls for more states. For these cases, $H(X)$ will be greater in value. If the distribution is skewed to favor certain states more, it indicates that specific states occur much more often. This means that the variety in states is much less, the system will see fewer states on the average, and $H(X)$ will have a generally smaller value than other cases. Defining complexity to be the variety of elements in the set results in greater values of $H(X)$ indicating that the complexity of the workspace is greater than the complexity of workspaces with smaller values of $H(X)$. In order to correctly model a more complex workspace, a system requires greater amounts of memory.

Related to $H(X)$ is another measure, *control entropy*. Control entropy characterizes the complexity of the mapping, δ. δ maps the set of states, X, to the set of controls, U, such that the total cost of executing a task is minimized. The mapping is stochastic in that there is a probability $Q_{u|x_i}$ that control u is chosen in state x_i. Let the collection of conditional probabilities be a matrix, Q. Each u in the solution then has an overall probability of occurrence

$$q_u = \sum_{x_i \in X} Q_{u|x_i} p(x_i) \tag{4}$$

With this definition of q, then control entropy is

$$H(U) = -\sum_{u \in U} q_u \log q_u \tag{5}$$

Recalling the one-to-one mapping Γ and the resulting equivalence of programs and controls ($p(\alpha) = p(u)$ for all $\alpha \in M$ and $u \in U$ such that $\Gamma(\alpha) \to u$), the control entropy is equal to the program entropy ($H(U) = H(M)$). This measure is the rate at which different controls are chosen to solve the problem.

Control entropy has one of two interpretations. First, $H(U)$ is the complexity of the solution. If the solution requires a greater number of unique controls, the q_u's will tend to be distributed more uniformly than the case when there are less programs (there will be a greater number of non-zero probabilities). For these cases, $H(U)$ tends to be greater and indicates that the solution is more complex. From this perspective, the greater $H(U)$ is, the more diverse the solution programs will be, thus indicating that more memory will be required to store the solution.

The second interpretation is that the entropy quantifies the uncertainty of the solution. If there is a choice of control for each state, then the q_u's can tend to be more evenly distributed. This distribution again leads to greater entropy values meaning that the solution has greater uncertainty due to the choice of controls.

This second view of entropy is not unlike Saridis' view[10] of the solution to problems for intelligent machines. He views the entropy of the solution as the measure of the uncertainty in the choice of controls for a given state. Entropy here is used more as an indication of the solution memory requirements and complexity of a workspace. Saridis' view, however, that the entropy of the solution should be minimized makes sense as the system should solve the problem as simply as possible.

The first two measures, state entropy and control entropy, deal with the memory requirements for the system and the complexity of the solution. They do not shed any light on the performance of the system with respect to any costs in the world. A third measure, *average work*, takes into account system performance.

It was mentioned above that a mapping exists from the set of states to the set of controls. The probability of choosing control u when in state x_i is $Q_{u|x_i}$. $f(x,u)$ determines the effect (next state encountered) of each u from a given x. Let the work for a system executing u from state x be $w(x,u)$, where $w(x,u) \geq 0$. This work can account for the effort exerted by the workspace in executing u, the amount of effort needed to reach the goal after executing u, or both. With these definitions, the average work done by a system is

$$\overline{W}(Q) = \sum_{x,u} p(x_i) Q_{u|x_i} w(x_i, u) \tag{6}$$

$\overline{W}(Q)$ is the expectation of the work over all states and controls, and it measures the average time, energy expended, path length, etc.. for a specific control policy. Whereas the entropy measures are measures of space requirements, average work is a measure of time or performance requirements.

Before proceeding to the next section, mention should be made of the interpretation of Q. It was stated above that Q is the matrix that specifies the mapping from the states to the associated controls. The individual probabilities determine the next choice of action for a state. It was also stated above that a policy is the system configuration along with the control algorithm that guides the system. Q taken together with the system resources defines the next action. Since the only difference among policies for a given set of resources is the control algorithm, Q can be interpreted as the policy.

4 Performance Bound

Entropy and average work each measure individual aspects of a navigation policy. In order to more completely characterize the performance of a policy, some combination of measures that quantifies both space and time is necessary. The chosen measures must be consistent for any policy and must accurately describe the solution.

A measure of space requirements for a policy needs to quantify the memory usage of the policy. It should provide information about the policy which is dependent in some way upon the policy. $H(X)$ and $H(U)$ are the two measures from which to choose. $H(X)$ is solely dependent upon the probabilities of occurrence of the states. For the current definition of the problem, it is independent of any specific policy and does not provide any information about specific policies. $H(U)$ is dependent upon some mapping Q, which is specific to a policy. $H(U)$ quantifies the complexity of the policy and the memory requirements of policy. Thus, control entropy is chosen over state entropy to characterize memory requirements of a policy.

The dual of space requirements is time requirements. The only measure that characterizes time requirements is the average work. Average work depends upon work measures $(w(x, u))$ associated with the application of controls. However, it is also dependent upon the policy as it is a function of Q. Therefore, it should produce a policy dependent time measure.

When the two measures are taken together, they characterize both the memory and effort requirements for a policy. The combination of these two measures is the *complexity-work characteristic* of a policy. The complexity-work characteristic is the ordered pair H-

$W = (H(U), \overline{W}(Q))$. Since H-W is independent of the specifics of a policy and depends only upon standardized quantities, its use allows comparison of different policies in a consistent manner.

As previously stated, different policies possess different complexity-work characteristics. Comparison of differing policies is possible, however, using their respective H-W's. The next obvious question is whether there is some policy which has the best complexity-work characteristic possible. Furthermore, if there is some best policy, can it be found computationally? The answer to both questions is yes and the following paragraphs provide the details for the answers.

Control policies map the set of states, X, into some set of controls, U. The mapping can be thought of as an encoding of the states into the associated controls. A measure that quantifies how well the mapping encodes the states is the *average mutual information* between the two sets.

Let the average mutual information between sets X and U be

$$I(X;U) \equiv I(p,Q) = \sum_{x,u} p(x_i) Q_{u|x_i} \log \frac{Q_{u|x_i}}{q_u} \tag{7}$$

This quantity is an indication of how much information one set contains about another set. AMI is a non-negative quantity and provides a lower bound to the entropies of the two sets whose relatedness it characterizes

$$\begin{aligned} 0 \le I(p,Q) \le H(X) \\ 0 \le I(p,Q) \le H(U) \end{aligned} \tag{8}$$

It is a measure of how well the controls provide information about the states subject to the mapping from X to U. If the choice of controls tends to be independent of the state, the mutual information tends to be smaller $((Q_{u|x_i} \approx q_u) \rightarrow I(p,Q) \approx 0))$. If the choice of controls is dependent on the state, AMI tends to be larger $((Q_{u|x_i} \neq q_u) \rightarrow I(p,Q) > 0))$.

AMI measures the correlation between the controls and the states. The greater the AMI, the better the encoding of states by the controls.

The concept of AMI together with average work define a lower bound on policy performance. Let the *complexity-work function* be

$$I(W) = \min_{Q \in Q_W} (I(Q)) \tag{9}$$

where Q_W is the set of all possible probabilities that achieve the level of work W. The complexity-work function is the minimum value of $I(Q)$ that is required to meet a certain level of work $\overline{W}(Q)$. It identifies the specific $Q_{u|x_i}$'s over the entire set of Q's which produce the given level of work.

The complexity-work function is the *optimal performance theoretically achievable* (OPTA) by a control policy. It defines the set of optimal policies ($Q_{u|x_i}$'s) in a workspace that satisfy different requirements on average work. Since AMI is the lower bound to the entropies of the sets it measures, the complexity-work function also indicates the minimum amount of resources that a policy requires to solve a specific control problem.

$I(W)$ is analogous to the *rate-distortion* function from source coding theory[5]. This function (also known as the Shannon Bound) has specific properties when the work function, $w(x, u)$, is chosen appropriately. $I(W)$ is a convex-downward, monotonically decreasing function. It is also invertible. That is, for some $\delta, \epsilon > 0$, if $I(\delta) = \epsilon$ then $W(\epsilon) = \delta$. This means that the function can be used to determine the minimum work associated with a given level of resources (controls) or the minimum amount of resources for a given level of work. Additionally, the optimal policy, in terms of the $Q_{u|x_i}$'s, is defined.

The existence of this bound stems from the work done in Information Theory by Shannon[11]. Later work by Blahut[3] describes an algorithm that actually computes the bound. The algorithm exploits the convexity of the entropies to solve the constrained optimization problem implied by (9). For a given set of $p(x_i)$'s and $w(x, u)$'s, it computes the value of the $Q_{u|x_i}$'s,

$I(Q)$, and $\overline{W}(Q)$ at various slopes. Formally, the algorithm is:

1. Begin with an initial estimate of the distribution for the controls. Let the estimate be $q_u^{(0)}$ and set $m = 0$,

2. Compute the optimal conditional probabilities $Q_{u|x_i}^{(m)}$ for $q_u^{(m)}$ by

$$Q_{u|x_i}^{(m)} = \frac{q_u^{(m)} \exp(s\ w(x_i, u))}{\sum_j q_j^{(m)} \exp(s\ w(x_i, j))}$$

   where $s < 0$ is the slope of the bound at the point of interest.

3. From the $Q_{u|x_i}^{(m)}$'s, determine the distribution of controls from the definition

$$q_u^{(m+1)} = \sum_{i=1}^{|X|} Q_{u|x_i}^{(m)} p(x_i)$$

4. Evaluate $I(p, Q^{(m)})$ and $\overline{W}(Q^{(m)})$ and check for convergence. If convergence requirements have been met, stop. Otherwise, set $m = m + 1$ and go to step 2.

The algorithm produces as much of the bound as necessary by repeating the steps for a variety of slopes.

The importance of Blahut's algorithm is twofold. It allows computation of the theoretically best performance that a policy can achieve. Designers can determine the memory requirements for a solution to a problem for a specific level of work or vice versa from the bound. For example, suppose a designer wants to ensure that a system will not exceed some maximum work requirement, δ. The amount of resources required to satisfy this requirement is found by locating the point on the Shannon bound that corresponds to $W = \delta$. The minimum complexity of the solution can be read directly from the bound $H(U) = I(W)|_{w=\delta}$. If the value is $I(W) = \epsilon$, then systems that have a complexity $H(M) \geq \epsilon$ will be the only ones capable of satisfying the requirements. The same sort of arguments can be used to determine the minimum possible work for a given maximum complexity value.

The second and probably more significant reason that Blahut's algorithm is important is that it actually computes the set of optimal navigation policies for a workspace subject to various system constraints. Q is the matrix that maps states to actions and can be considered as equivalent to the policy. Blahut's algorithm computes the optimal Q's for various values of work and complexity. By the nature of the algorithm and the bound, the Q's that are computed are optimal with respect to the workspace.

5 Simple Navigation Problem

A simple navigation problem serves as an illustration for the concepts presented and their use. A mobile system navigates a simple world to reach some goal. The mobile system has sensory equipment adequate to identify the states of the world uniquely ($\mathcal{P} = X$.) For simplicity, the system maps the world states into a binary valued vector. A state x is mapped into the vector $x_i = (0, \ldots, 0, 1, 0, \ldots, 0)$, where the 1 is in the i_{th} position. The system also has adequate memory capacity to store the states of the workspace.

The mobile system implements four navigation policies. Three of the policies are directed search policies while the fourth is an undirected search policy. The specific policies are a standard breadth-first search, a random walk, a counter-based search, and an A^* search. The policies choose a motor program to execute which is dependent upon the individual algorithms. The set of motor programs from which to choose is a set of basic actions: WEST, NORTH, EAST, and SOUTH.

A grid world is the workspace in which the system navigates. It is a set of cells which are interconnected. Two cells are said to be adjacent if they share a common edge. A cell can have adjacent cells in any of the four compass headings: WEST, NORTH, EAST, and SOUTH. Movement is only allowed between adjacent cells.

Additional properties of the grid world are as follows. A single goal state (cell) exists and is called g. This means that $X_F = \{g\}$. The costs of the motor programs, $w(x, u)$, are uniformly distributed in the range $[0.95, 1.05]$. The distribution of the states is uniform with $p(x_i) = \frac{1}{|X|}$. The work associated with the application of motor program u from state x_i is

$$w(x,u) = c(x,u) + J^*(f(x,u)) \tag{10}$$

where J^* is the *optimal return function*[4]. The work indicates the cost of executing a motor program and then assuming that an optimal path is followed from the succeeding state. It is this work that is used in the computation of the theoretical bound.

Figure 2 shows the grid world that this example uses. The "$*$" indicates the goal. Figure 3 presents the Shannon Bound for this world. Notice that the minimum value of work displayed on the plot is zero. This is because a bias was subtracted from the work to bring the value to zero. This can be done because it can be assumed that for each state there will be an action which has minimum work for that state ($w_{\min}(x_i) = \min_u(w(x_i, u))$.) The minimum overall work is found when $Q_{u|x_i} = 1$ for $w(x_i, u) = w_{\min}(x_i)$ and $Q_{u|x_i} = 0$ for $w(x_i, u) \neq w_{\min}(x_i)$. The minimum work is then

$$\overline{W}_{\min} = \sum_{i=1}^{|X|} p(x_i) w_{\min}(x_i) \tag{11}$$

The experiment consists of two parts, the first of which is evaluating each of the four navigation policies. For each policy, the system undergoes a series of trials. Each trial consists of randomly placing the system in a state of the workspace using the $p(x_i)$'s. After placement, the system uses the current navigation policy to traverse the world to the goal. During the traversal to the goal, the system monitors the number of times each action is executed in each state and forms the $Q_{u|x_i}$ probabilities empirically. At the same time, the system notes the amount of work done for each trial. After the total number of trials for each policy, the composite Q's and average work is computed and stored. It is these values that are used to compare the policies with each other and with the bound.

The four policies are examples of policies that are used in performing searches of workspaces. The breadth-first search (BFS)[1] is a standard graph search technique from computer science. It performs a directed, exhaustive search of the graph until the goal is found. Due to the fact that it must backtrack quite often in reaching its next state, it would be expected to have a large average work. Also, it would be expected to have a low to mid-range complexity as it explores each possible action for a state (meaning it produces a mapping from states to actions that tends toward uniformity.)

A random search forms the basis for the first iteration of many reinforcement learning algorithms[13]. The first iteration of these learning techniques degenerates into a simple random walk. This policy is an undirected policy as the choice of actions does not depend upon the state. It should have very high work and very low complexity.

Counter-based searches (CB) are proposed by Thrun[13]. Their use is mainly in reinforcement learning techniques but they are directed searches. Thrun suggests that these are very efficient policies, thus their inclusion in the experiment. They are expected to have low work; and due to initial exhaustive exploration of the states, they should have low to mid-range complexity.

A^* is a heuristic search developed by Hart[6]. It uses a heuristic function to guide its choice of the next state to explore. Unlike the rest, it does not have to explore the entire workspace. It uses additional information to guide its search. It is expected to have low work and a complexity tending toward the mid-range.

Experimental results agree very well with what was expected, and they are displayed in Figure 4. The ordering among the policies with regard to work is as expected with the random policy being the worst, the BFS being somewhat better, the CB search being better still, and the A^* being the best.

The ordering among the complexities of the solutions is not surprising. The random search has a complexity of zero as the choice of actions is independent of any state. The BFS and CB searches have choices of actions that are not totally independent of the state, but still have a significant amount of purely random choice. Thus, their complexities are somewhat larger than the random search. A^* has the greatest complexity as the choice of actions is guided by a heuristic. The actions for each state should be well defined if the heuristic describes the workspace well which means that the choice of actions should not be uniformly distributed for each state.

The second part of the experiment is the computation and use of supposed optimal policies using Blahut's algorithm. Various values of policy complexity ($H(U) = 1.3, 1.0, 0.5$) are chosen and the optimal policy is computed for these values. Navigation experiments use the Q's generated by the algorithm and monitor the amount of work needed to solve the problem. The implementation of the experiments is the same as that for the previous navigation policies except that the Q's are used instead of an explicit control algorithm. Figure 5 presents a control architecture that implements the Q's using a selection agent that moderates an ensemble of output agents. The selection agent is a stochastic switch that uses the $Q_{u|x_i}$'s, while each output agent has a fixed control for its output.

Figure 6 presents the results. It is obvious that the computed policies are not in all cases the optimal policy. For the most complex policy, the solution is very near optimal. As the complexity of the solution decreases, the associated work increases faster than the bound would indicate. This is due to the choice of the work measure, $w(\alpha, i) = c(\alpha, i) + J^*(\hat{\delta}(\alpha, i))$. The measure assumes that the optimal path is followed after the application of the control program, which is not the case. Future work will address this problem by attempting to define a work measure that better approximates the amount of work that is *expected* to be performed after executing a motor program. However, the bound that is presented here will still be a lower bound in that it is the best that can possibly be done.

6 Discussion

The concepts presented in this paper describe the complexity of the solutions to control problems with respect to the amount of work that is performed using the solutions. The methods presented are an alternative and extension to the work of Saridis and Valavanis, who consider the minimization of solution complexity for optimal problems. While this problem is important, their method does not explicitly address the solution in cases where sub-optimal performance may be acceptable in light of constraints on system time and space resources.

The methods presented here use source coding approaches (entropy and work measures) to describe the solution complexity and performance of a system. The approaches are not yet complete as evidenced by the results of the optimal policy evaluation. The work measure presented is not necessarily appropriate as the policy computed from Blahut's algorithm should exhibit a work value that meets the bound. The difference in this value is most likely due to the fact that the work measure used assumes that an optimal path is followed after the execution of a control. Due to the random nature of the choice of actions, this will obviously not be the case. Redefinition or reinterpretation of the measure is necessary.

The probabilities that individual states occur is also not necessarily accurate. The paper assumes something of a "snapshot" of the states to get their probability of occurrence. However, it is obvious that the probability of occurrence will not only depend upon the number of times a state occurs in the FA. The probability will also depend upon how often the state is *encountered* by a policy. Therefore, adjustments to the state probabilities need to be made to account for the action of individual policies. These adjustments might be made using some sort of iterative algorithm.

The difficulties presented by both the work and state probabilities suggest that the computation of optimal policies is not as straightforward as was originally thought. The experimental results that evaluate the optimal policies tend to support this view. The suggestion that some sort of iterative approach be used to update the state probabilities hints at the idea that an iterative approach may be suitable for the computation of optimal policies. Such an approach would yield optimal policies that meet both complexity and work requirements.

Finally, this paper presents a consistent framework for studying the performance and complexity of solutions to optimal control problems. A well defined body of work exists in the form of information theory and source coding theory to support the analysis. The quantities presented provide consistent measures for describing and comparing control policies.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1983.

[2] A. G. Barto, R. S. Sutton, and C.J.C.H. Watkins. Learning and sequential decision making. Technical Report 89–95, COINS, September 1989.

[3] R. E. Blahut. Computation of channel capacity and rate-distortion functions. *IEEE Transactions on Information Theory*, IT-18(4):460–473, July 1972.

[4] A. E. Bryson, Jr. and Y. C. Ho. *Applied Optimal Control: Optimization, Estimation, and Control*. Hemisphere Publishing Corporation, New York, 1975.

[5] R. M. Gray. *Source Coding Theory*. Kluwer Academic Publishers, Boston, Massachusetts, 1990.

[6] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions of Systems Science and Cybernetics*, SSC-4(2):100–107, July 1968.

[7] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1979.

[8] M. D. Lemmon. Real-time optimal path planning using a distributed computing paradigm. In *Proceedings of the American Control Conference*, pages 2062–2063, Boston, Massachusetts, 1991. American Automatic Control Council.

[9] G. N. Saridis. Entropy formulation of optimal and adaptive control. *IEEE Transactions on Automatic Control*, 33(8):713–721, August 1988.

[10] G. N. Saridis and K. P. Valavanis. Analytical design of intelligent machines. *Automatica*, 24(2):123–133, 1988.

[11] C. E. Shannon. A mathematical theory of communication. *Bell Systems Technical Journal*, 27:379–423,623–656, 1948.

[12] J. A. Stiver and P. J. Antsaklis. Modeling and analysis of hybrid control systems. In *Proceedings of the IEEE Conference on Decision and Control*, volume 4, pages 3748–3751, Tucson, Arizona, 1992.

[13] S. B. Thrun. Efficient exploration in reinforcement learning. Technical Report CMU-CS-92-102, School of Computer Science, Carnegie-Mellon University, January 1992.

[14] K. P. Valavanis and G. N. Saridis. Information-theoretic modeling of intelligent robotic systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(6):852–872, 1988.

[15] K. P. Valavanis and G. N. Saridis. Probabilistic modeling of intelligent robotic systems. *IEEE Transactions on Robotics and Automation*, 7(1):164–171, February 1991.

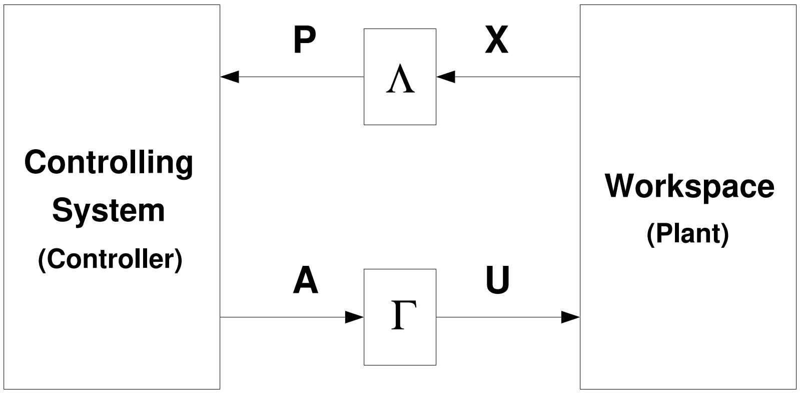

Figure 1: Model of System and Workspace

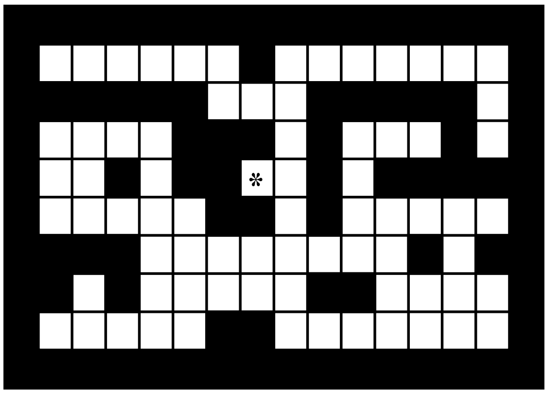

Figure 2: Grid Workspace

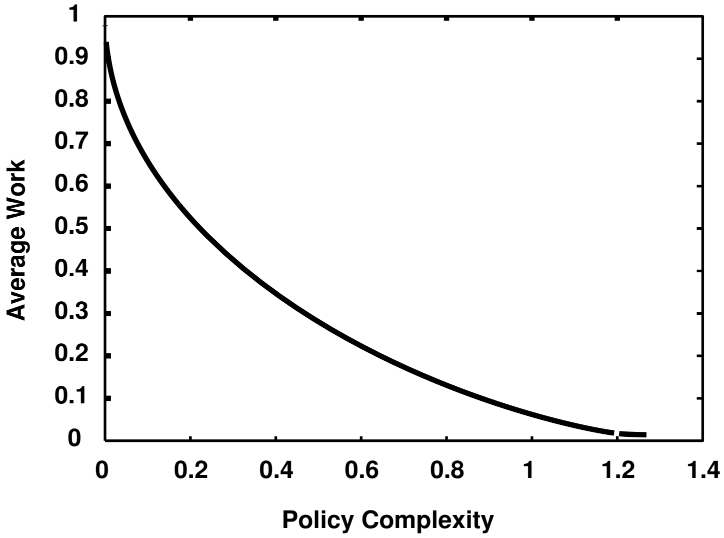

Figure 3: Shannon Bound

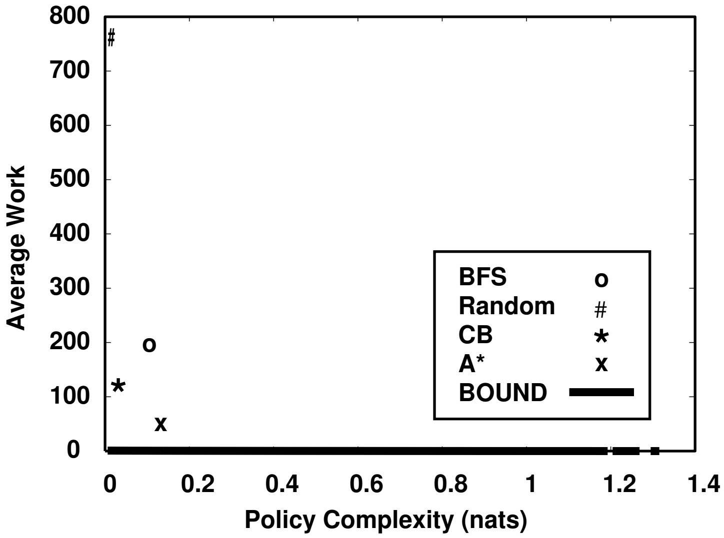

Figure 4: Experimental Policy Evaluations

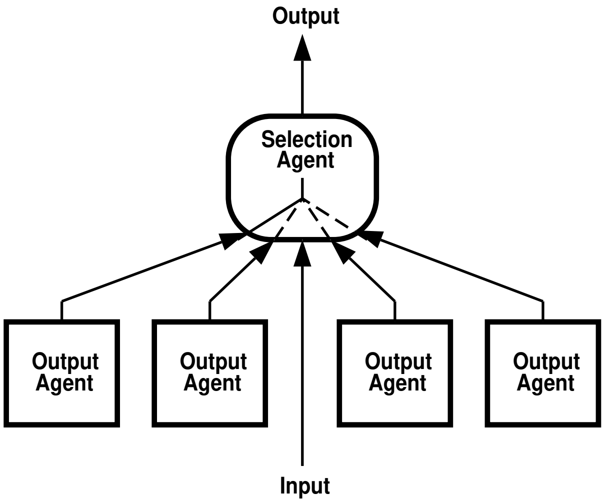

Figure 5: Stochastic Control Network

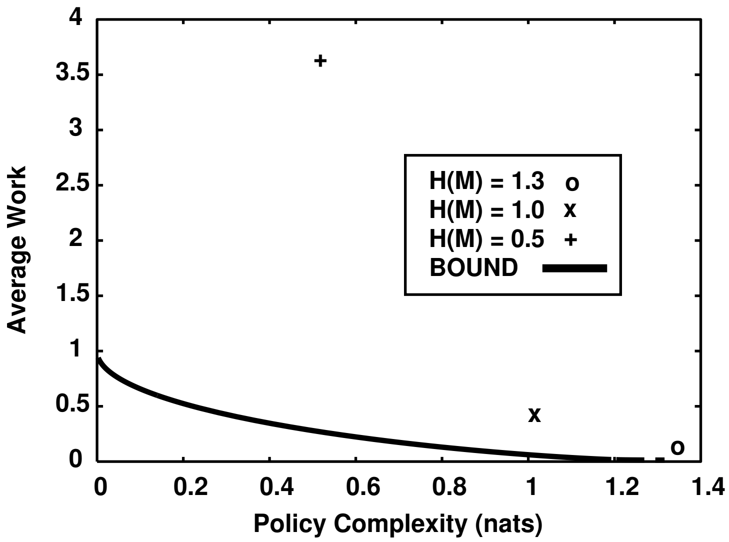

Figure 6: Optimal Policy Evaluations