Inductive Biases in a Reinforcement Learner

Helen G. Cobb
Naval Research Laboratory, Code 5514
Navy Center for Applied Research in Artificial Intelligence
4555 Overlook Ave., S.W., Washington, D.C. 20375-5320

DTIC
ELECTE
MAY 12 1995
C

Abstract

Reinforcement Learning Methods (RLMs) typically select candidate solutions stochastically based on a credibility space of hypotheses which the RLM maintains, either implicitly or explicitly. RLMs typically have both inductive and deductive aspects: they inductively improve their credibility space on a stage-by stage basis; they deductively select an appropriate response to incoming stimuli using their credibility space. In this sense, RLMs share some learning attributes in common with active, incremental concept learners. Unlike some concept learners that employ deterministic procedures for selecting hypotheses, however, the evaluations of hypotheses provided to RLMs are often uncertain, either due to noisy environments, or due to summary evaluations which occur after a sequence of learner-environment interactions. This paper examines issues of inductive learning bias in this context experimentally. Specifically, the paper addresses inductive learning biases in the context of a simple RLM called a Collective Learning Automaton (CLA). The CLA learns the shortest path through a small network. The research points out some of the difficulties of finding performance measures that indicate the strongest, correct biases for the automaton.

1 INTRODUCTION

Over the past few years, there has been a growing interest in the effects of bias in learning algorithms. In inductive concept learning, Mitchell considers bias to be the expressed preference of the learner for considering one hypothesis of a classification rule (or a generalization rule) over another (Mitchell, 1980). Mitchell points out that "an unbiased learning system's ability to classify new instances is no better than if it simply stored all the training instances and performed a table lookup when asked to classify a subsequent instance" (Mitchell, 1980, pp. 1). Bias, as defined in this sense, is necessary for any inductive learning algorithm.

Within concept learning algorithms there are two fundamental types of bias: **language bias** and **procedural bias**[1] (Utgoff, 1986; Rendell, 1986; Gordon, 1990).[2] Language biases are preferences which determine the expression of hypotheses of the target concept. Procedural biases are preferences which affect the traversing of the search space. Procedural biases also include halting rules. Both language biases and procedural biases can affect a learning algorithm's speed for finding a classification rule that is close to the target concept. The language bias reduces the size of the search space by constraining the number of possible formulations of the hypothesis; the procedural bias reduces the amount of traversal through the search space by constraining the search method.

Closely associated with notion of inductive bias are the notions of **strength** and **correctness** (Utgoff, 1986). To date, the definition of strength refers more to language bias than procedural bias. The strength of a language bias corresponds to the entire size of the hypothesis space that the learner can generate given no constraining procedural bias (e.g., exhaustive search with labeled instances). The strength of the language bias can be increased either by reducing the size of the representational space or by transforming the grammar to one that is less expressive. Not all hypotheses that may be generated from a language description may be acceptable hypotheses of the target concept, however. Language biases that permit the forma-

[1] Procedural bias is also called **algorithmic bias.**

[2] The biases within the problem space may be different from those within the algorithm. For example, instances examined by the learner may be expressed in a language which is much richer than the learner's language.

19950510 099

tion of hypotheses that do describe the target concept are called *correct* biases.[3]

In a general sense, procedural bias constrains the breadth, the direction, and the duration of the search for the classification rule. A strong procedural bias minimizes this trajectory through the search space. An incorrect procedural bias over-constrains or under-constrains this search so that learner cannot find a correct classification rule. A powerful concept learner is one that uses as strong a bias as possible, for both its language and procedural biases, without sacrificing correctness.

Since many learning algorithms have an inductive component, the analysis of learning in terms of language and procedural bias applies broadly to many algorithms other than traditional concept learners. For example, a large number of active incremental learners, which are generally not considered to be traditional concept learners employ what is referred to as **Reinforcement Learning Methods (RLMs)** (Whitehead and Ballard, 1990). RLMs have an inductive component in that they constantly revise the credibilities of their hypotheses based on experience. These learners then deduce a response from an input stimulus based on their current hypothesis or set of hypotheses. RLMs are often used to learn first-order decision *sequences* in dynamical systems. Such algorithms include genetic algorithm-based systems (Holland, 1975, De Jong, 1975) such as Grefenstette's SAMUEL (Grefenstette, 1988), Artificial Neural Networks (ANNs) (Rumelhart, 1986) such as those consisting of Widrow's ADALINE units (Widrow, 1985), and Q-Learners (Watkins, 1989). This paper, in particular, examines a simple example of a RLM called a **Collective Learning Automaton (CLA)** (Bock 1992).

Viewing RLMs as having both inductive and deductive components is supported by other researchers, including those studying learning automata (Narendra and Thathachar, 1989) and those working in information theoretics as applied to inductive and deductive inference (e.g., see Watanabe, 1960). Furthermore, these researchers believe that both forms of inference are necessary for learning: "Inductive and deductive inference do not contradict but merely complement each other and both are found to be essential for learning processes" (Narendra and Thathachar, 1989, pp. 15). Watanabe states: "Inductive inference contains, as a necessary ingredient, a constant comparison of the deductive consequence from a hypothesis with the experiment. Accordingly, the model theory of inductive inference must permit deductive inference to play a corresponding role within its framework" (Watanabe, 1960, pp. 208).

[3] In the PAC learning framework (Valiant, 1984), hypotheses describe the target concept within limits of accuracy specified as part of the procedural bias).

A RLM is similar to an active incremental concept learner that is inferring a single concept. The purpose of the concept learner is to find a final hypothesis sufficiently close to the target hypothesis that *covers* the positive instances the learner has observed. The purpose of a typical RLM is to seek one solution (or a few) to a decision sequence problem. In the latter case, the final hypothesis covers only a small fraction of the possible sequences. If the space being covered is very small and the instances being received only have the label of "positive" or "negative," then any inference procedure, deductive or inductive, would probably do no better than random search. Instead, RLMs usually receive evaluations which measure, in a sense, the "degree of positiveness," (or better, the "degree of credibility") of an entire solution (or trial hypothesis) which can be thought of as a group of instances. A simple analogous problem for a concept learner would be to generate the set of positive instances covered by its current hypothesis for evaluation. A summary score might indicate the actual fraction of positive instances in the set. A difference between the concept learner and the RLM in this case is that the RLM examines a sequential relationship among the decisions it generates whereas the concept learner applies a "same member of" type of relationship.

In the RLM, each individual decision is not directly evaluated as "good" or "bad." The problem for the RLM then becomes one of inferring what makes up the constituent parts of the best solution (decisions in the final hypothesis) from summary evaluations of trial hypotheses, rather than trying to infer a final hypothesis through the direct evaluation of constituent members which are labelled as "positive" or "negative" instances. This inference problem of finding out the constituent parts of the best solution is commonly referred to as the credit assignment problem.

There are basically two approaches for studying biases in learning systems. One approach is to maintain a set of biases while the learner performs a task, and then before attempting the task again, search for a better set based on the learner's overall performance. The other approach is to dynamically adjust a set of biases during the learning task based on intermediate evaluations of performance. This paper calls the first approach **inherited bias** (or fixed bias). The second approach is referred to in the literature as **dynamic bias adjustment** (or shift of bias) (Gordon, 1990; Rendell, 1987; Schlimmer, 1987; Utgoff, 1982, 1986). Both of these approaches have biological analogs. For example, the range of frequencies that a creature can hear is usually a genetically determined trait that does not improve due to learning over the creature's life-span. Not all characteristics fit this category, however. A creature having a fairly long life-span must be able to dynamically adjust its initial biases in order to adapt itself to changing circumstances. For example, biases appropriate to one stage of development, may be inappropriate at another. So

ion For
CRA&I
TAB
nounced
ification
ibution /
Availability C
Dist | Avail and / Special
A-1

we can think of the dynamic shifting of bias as analogous to bias changes that occur during various stages of individual development.

Since of the primary objective of this research is to examine the effects of learning biases, and their interactions, on the performance of a RLM, this study focuses on examining the effects of inherited biases. Being able to compare the effect of making one biasing assumption over another is possible when the combination of selected biases remains constant over a learning task. Using dynamic bias adjustment involves additional issues. One major issue that needs to be addressed is how to modify the state of the learner so that it can continue to learn with a new set of biases. This issue is complicated by the fact that most RLMs do not maintain an episodic memory of their experiences, whereas many concept learners save instance information so that biases can be adjusted retroactively, if necessary, using backtracking. There are so many ways of implementing dynamic bias adjustment that discovering the best method for a RLM is a study unto itself.

There are several questions of interest in studying inherited biases in RLMs, assuming that a learning task represents the life-span of the RLM. For example, what are characteristic language and procedural biases in the RLM? If we decompose the problem into two subsystems, one subsystem for modeling the learner, and the other subsystem for selecting the inherited biases, how should the two subsystems be designed? What performance measure(s) should be sent from the learning subsystem to the bias search subsystem which assigns a the learner it's inherited biases? Are there any interactions between the procedural and representational biases? This work represents some preliminary results in studying combinations of biases in a simple RLM without examining the bias search subsystem.

The next section provides background information, introducing terminology that frames the question of inductive biases in the context of RLMs. This section also discusses a way of viewing the strength of a bias in empirical terms as information compression. Section 3 gives a quick overview of the CLA; Section 4 describes the particular shortest path problem being examined, including an outline of the experimental design; Section 5 presents the experimental results, and Section 6 gives a brief conclusion.

2 INDUCTIVE BIASES

2.1 MEASURING BIAS STRENGTH

The definition of strength of a language bias given at the beginning of the introduction permits us to perform some analytic evaluation of the limiting performance of a concept learner. However, if we automate the search for the strongest, correct bias we would like to find some performance measures which would empirically allow us to evaluate biasing assumptions. This is especially true if we consider the problem of choosing a good procedural bias.

In RLMs, it is sometimes useful to examine a component of language bias called **representational bias** (Gordon, 1990; Rendell, 1987; Schlimmer, 1987; Utgoff, 1982, 1986; Mitchell, 1980). Representational bias defines the choice of atoms or primitives in the hypothesis language.Very often, the grammar for expressing hypotheses is defined in the paradigm of the RLM. For example, a RLM may always generates a decision sequence, or trial hypothesis, having a specified format. A trial hypothesis is generated from the RLM's memory. This memory could be expressed as a linear expression, a set of productions, or perhaps a state transition table. The choice of terms used in the RLM's memory represents a representational bias within the learner. If we examine a problem in which the best representational bias is known, then we can investigate which performance measures can be used to measure a strong, correct representational bias.

A major procedural bias within RLMs is the amount of change made to the memory at each stage of learning. Several authors (e.g., Sutton, 1989) call this factor the learning rate.[4] The best setting of this factor cannot usually be determined from the problem definition. However, by examining various levels of this factor in conjunction with different representations, we can test the effect of this bias on selected performance measures.

2.2 INDUCTIVE COMPRESSION

Generally speaking, inductive learners compress information. Watanabe points out that there are several steps involved in the information compression (Watanabe, 1971). Using different terms than Watanabe's, we can consider the choice of representation of the language as the first compression step, the grammar of the language as the next compression step, and finally, the inductive compression specified by the procedure as the third compression step.

Representation -> Grammar -> Inductive Procedure

Watanabe also points out that there are several ways to arrive at the same final level of compression. For example, all the burden can be placed on initially compressing information into a single concept class, in which case a grammar is not needed. In this case, all of the effort is applied to finding the best representation for the problem. Alterna-

[4] This term learning rate is not used in this paper because many factors actually affect the rate of learning.

tively, very little compression can be used in developing primitives, in which case more burden is placed on developing a grammar and applying an inductive inference process. In this second case, the resulting classification rule may be more complex (e.g., more terms) than the case where more effort is expended in finding a good representation. Watanabe speculates that the most desirable situation is to have a "well-balanced distribution of information compression over different steps" (Watanabe, 1971, pp. 567).

2.2.1 Language Bias

Notice that the first two steps correspond to defining a language bias. When we select a language bias, we are in effect compressing information. Unless the language bias is chosen either arbitrarily or from the experience of the designer of the learning algorithm, a number of steps are required to compress raw data into a language specification. We can define the strength of a language bias in terms of the amount information compression that occurs over these steps. However, most analyses of algorithms do not consider the cost of this phase; that is, an initial language bias is often assumed analytically to be a "given." Thus, assuming that we can perform compression instantaneously, then we can define the strength of a language bias as follows: *If a language bias, A, performs more information compression over the same set of data than a language bias, B, then A is a stronger language bias than B.*

This definition is an operational extension of the original definition of strength of a language bias which refers to the size of the hypothesis space. Given the same raw data, a language specification A, results in a smaller hypothesis space than some other language specification B, if and only if the language A performs more information compression of the raw data than language B. It follows that the bias of A is stronger than the bias of B.

Notice that the strength of a bias is relative to the original data. Suppose that we start with two different sets of raw data, set A, and set B which is an elaboration of set A, and we arrive at the exact same language specification, L (i.e., $L_A = L_B$) for both sets. The strengths of the language biases are not the same even though the sizes of the hypotheses spaces are. The second specification compresses more information relative to the original data set B.

Also notice that this definition ignores the notion of correctness, which requires some target concept or problem definition. For example, when we compress the raw data sets A and B to the same language (as in the last example), the compression for B may be greater, but the compression may also be incorrect (i.e., too strong).

2.2.2 Procedural Bias

To date, little attention has been directed toward the definition of the strength of a procedural bias. We extend the definition of language bias to define the strength of a procedural bias. Since inference procedures usually occur over several steps, we can define the strength of a procedural bias in terms of its rate of information compression: *If a procedural bias A takes a fewer number of stages to perform the same amount of compression as procedural bias B, then procedural bias A is stronger than procedural bias B.* Alternatively: *If a procedural bias A compresses more information in the same number of steps as procedural bias B, then procedural bias A is stronger than procedural bias B.*

Procedural bias essentially differs from language bias in that shifting a language bias usually implies uniform changes throughout the learning process. For example, dropping a term from the language must be done in all hypotheses and all logic leading to those hypotheses. Changes in procedural bias are done in the context of efficiently finding a concept description or a problem solution given some language bias constraint.

2.3 A RLM'S CREDIBILITY SPACE

A concept learner that receives labelled instances from a generator can, in principle, maintain a **version space** (Mitchell, 1977) of hypotheses that are consistent with all of the instances seen so far. For example, Mitchell maintains in his Candidate Elimination Algorithm a maximally specific set and a maximally general set of hypotheses so that not all hypotheses need to be considered explicitly. As learning progresses, the version space becomes smaller. Haussler points out that one of the problems with Mitchell's approach is that the storage space for this set of hypotheses can still increase exponentially in size. Haussler's analysis shows that it is not necessary for a concept learner to maintain an explicit version space at all (Haussler, 1987). By examining a sufficient number of instances, the learner can develop a hypothesis that becomes ε-close to the target concept with some large probability $1 - \delta$.

One of the important assumptions behind Haussler's analysis, however, is that the learner can reject a hypothesis because all of the instances are typically labelled as positive or negative *with certainty.* If the learner develops a hypothesis that is consistent with all instances seen and it turns out that instance information is occasionally incorrect due to mislabelling, then the resulting classification rule may effectively over-fit the instance data to include noise (Spears and Gordon, 1992). In a learning situation where an instance is only labelled as to how probable an example it is of the target concept (i.e., a probabilistic

learner), maintaining a version space of consistent hypotheses in some deterministic sense loses its meaning.

In RLMs, a version space instead consists of a single probabilistic hypothesis that covers the space of all possible hypotheses. Equivalently, we can consider the version space to be set of all hypotheses, with each hypothesis given a value indicating its *degree of its credibility* relative to other hypotheses. Let us call this probabilistic version space a **credibility space,**[5] and for convenience, let us call the degree of certainty of a hypothesis its **credibility value.** Credibility values may be decision weights, rule strengths, or transition probabilities, depending upon the RLM.

In concept learners, a hypothesis induces a dichotomy of the instance space (Haussler, 1988). Haussler defines the growth function as the maximum number of dichotomies that can be induced in a hypothesis space for a finite number of instances. He then shows that because the growth function can be related to the number of instances required to ε-exhaust a version space, it can be used as a measure of bias strength in a probabilistic sense (Haussler, 1988, pp. 191). The analogy to the dichotomies of instances in concept learning is the number of inputs and outputs available to a RLM over a decision sequence. Even though this information does give us a bound on the size of the hypothesis space, unfortunately, it does not give us a direct bound the amount of testing required. Because RLMs often use non-binary evaluations, a RLM must consider a trial hypothesis repeatedly to gain confidence in its credibility. It is very unlikely that the RLM can reduce the credibility space by testing a hypothesis once; however, testing does allow the RLM to reshape the credibility values so that retesting of poor hypotheses is minimized.

2.4 INFORMATION COMPRESSION AND ENTROPY

After each stage of the search, a RLM adjusts its credibility values to reflect the outcome of its experience. Depending on the policies employed by the RLM, these values are increased or decreased so that future decisions should provide better overall evaluations. The update policy generally indicates the rate of change in the credibility values (i.e., amount of change per stage). Depending on this rate, the credibility space of the hypotheses becomes organized more or less quickly.

A measure of the amount of compression per stage is the change in the entropy. We can compute the entropy at a stage using Shannon's entropy measure (Shannon, 1948, pp. 393)

$$H_{stage} = -K\sum_{i=1}^{N} p_i \log(p_i), \qquad \text{(Eqn 1)}$$

where K is a positive constant, and N is the current size of the credibility space at the current stage. Let us assume for convenience that K is 1.

According to Watanabe: "Inductive inference is a process such that the distribution of weights (credibilities) becomes increasingly concentrated on a decreasing number of cases (hypotheses) no matter how widely one distributes the weights initially" (Watanabe, 1960, pp. 210). Watanabe calls this observed decrease in the entropy the **inverse H-theorem** (Watanabe, 1960, 1975).

The principle of decreasing entropy applies to concept learners. Suppose that we start with a hypothesis space having cardinality $|H|$ and there is an equal probability of considering each of the hypotheses. If we can eliminate hypotheses due to having certain instance information, and of there is an equal probability of inspecting the remaining hypotheses, then the entropy in the version space after eliminating all but N hypotheses is

$$H = -\sum_{i=1}^{N} (1/N)\log(1/N) \ . \qquad \text{(Eqn 2)}$$

This term simply reduces to $-\log N$. The quantity in (Eqn 2) can be interpreted as the "amount of ignorance" of not knowing which of the N hypotheses is correct (Watanabe, 561, pp. 562). When $N = 1$, the uncertainty reduces to zero.

In concept learners, the probabilities are not actually the same for the hypotheses. Some hypotheses are given more weight than others depending on the procedural biases. Because many procedural biases only implicitly generate a distribution over hypotheses in the version space, computing entropy for these cases becomes a challenge. In a RLM such as a CLA, it is possible to calculate entropy since transition probabilities can be used to compute the credibility values of trial hypotheses. These credibility values can be used as probabilities in (Eqn 1).

3 COLLECTIVE LEARNING AUTOMATON (CLA)

3.1 OVERVIEW

The CLA is an iterative paradigm that refines its hypotheses of the solution at each stage of the search. Within each

[5] Both Rendell (Rendell, 1986) and Watanabe (Watanabe, 1960) use similar terminology. Rendell defines a **credibility function** of hypotheses which assesses the credibility or belief of the various competing hypotheses.

stage, the automaton communicates with the environment for several **interactions.** For each interaction, the automaton receives a stimulus input from the environment, selects a response output, collects the stimulus-response pair into a history, and then transmits the output to the environment. This interaction cycle repeats until the automaton generates a **sequence** of stimulus-response interactions called a **collection.** At the end of a stage, the environment transmits an **evaluation** to the automaton. The automaton is *collective* because the evaluation of the decision sequence does not occur until a collection of stimulus-response pairs are obtained.[6]

The CLA maintains a **State Transition Matrix (STM).** The STM explicitly provides stimulus-response probabilities by partitioning the stimulus space into a discrete number of compartments called **stimulants** and the response space into a discrete number of compartments called **respondents.** The sum of the probabilities across respondents for a given stimulant is one. The automaton applies a **selection function** to choose a response to an environmental stimulus based on the current contents of the STM. To modify the STM, the automaton first develops a **compensation** based on an internal transformation of the evaluation; then the automaton's **update function** changes the STM probabilities using both the compensation and the stimulus-response information stored in the history. The cycle repeats for several stages until some convergence criterion is met. Figure 1 summarizes the steps using highlighted pseudocode.

The simplicity of the CLA makes it potentially amenable to analysis. Because the probabilities of a respondent is given for each stimulant, it is possible to compute an estimate of the entropy at each stage. This estimate is called the **collection entropy,** H_c. To calculate the entropy, the automaton first computes, on the fly, the product of the conditional probabilities of selecting responses for all interactions except for the last. For a collection of length l, the path probability, p_{path}, is

$$p_{path} = \prod_{m=1}^{l-1} p_m. \qquad \text{(Eqn 3)}$$

The automaton then uses this probability in computing H_c at the final interaction:

$$H_c = \sum_{j=1}^{r} (p_{path} p_j) \log (p_{path} p_j) \qquad \text{(Eqn 4)}$$

Some other RLMs, such as neural networks, *implicitly* represent the relationship between stimulants and respondents by using a linear expression which maps the weighted sum of the stimuli into a response. Based on evaluation feedback, these learners adjust the weights within the expression to modify the associated response. Still other RLMs use stimulus-response rules that permit condition parts of rules to intersect. This intersection corresponds to having *overlapping stimulants.* Many of these RLMs are interesting paradigms; CLA has the virtue of having a simple automaton underpinning in which the probabilities of input-output associations are explicitly enumerated. The results of this study may be useful in examining other RLMs.

[6] Sutton calls problems which restrict reinforcement to occur at the end of a sequence "time blinded tasks," because the reinforcement time interval is very often unknown (Sutton, 1984).

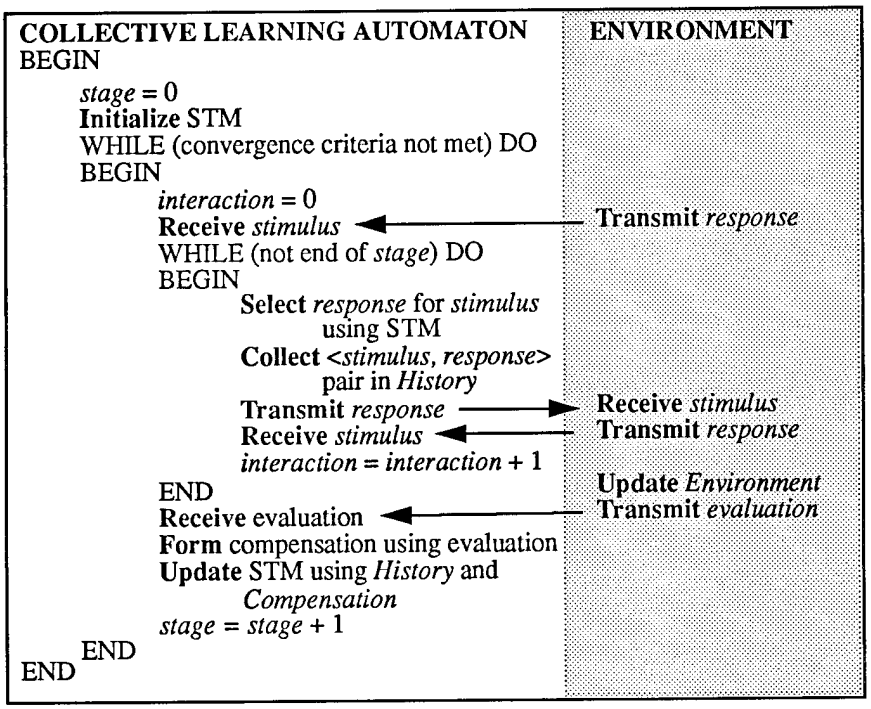

Figure 1: A Standard Collective Learning Automaton

3.2 LEARNING BIASES IN THE CLA

Now let us briefly consider examples of biases within the CLA. A example of a representational bias in the automaton is the definition of the states in the STM (i.e., the stimulants and the respondents). Expressing a hypothesis as a decision sequence is an example of a grammar bias. Thus, the language of a trial hypothesis consists of a sequence of stimulant-respondent ordered pairs. There are several examples of procedural bias. The formulation of the compensation and update functions reflect inductive procedural biases. Deductive procedural biases are reflected in the selection function. For example, the automaton may make each decision based only on current state information without looking behind at previous information (i.e., the automaton may be first-order). The selection function governing how the automaton chooses a decision sequence (i.e., a hypothesis) from the STM may consider all hypotheses or only a subset of hypotheses whose probabilities lie above a threshold.

4 EXPERIMENTAL APPROACH

4.1 AN EXAMPLE PROBLEM

For purposes of illustration, let us examine a small shortest path problem. Figure 2 below summarizes the environment's complete knowledge of the problem. The stimulants that the automaton can receive are the colors within the ellipses. The lines shaded in grey show the optimal path. The respondents are X, Y, and Z; the time of travel associated with each response is placed in parentheses.

Notice that it is possible for more than one respondent to lead to the same next stimulant. For example, for the stimulant called VIOLET, all three of the respondents, X, Y, and Z, lead to the stimulant called RED. The time travelled depends on the actual respondent selected. Also notice that in this problem, the shortest path in terms of time is the longest route through the network in terms of the number of decisions required (i.e., five decisions).

It is interesting to note that there is an inherent hypothesis procedural bias in the automaton because it selects responses on a node-by-node basis. Even when the probabilities of selecting different *responses* are initially the same at each of the nodes, the probabilities of selecting the possible action *sequences* are different. There are more paths leading to the RED stimulus if the initial response is X (7 paths) than if the initial response is either Y (2 paths) or Z (4 paths), so that the paths starting with an X response are initially explored less than the paths starting with a Y or Z responses. This bias is compounded if there are near-optimal solutions in the more easily explored parts of the search space. In Figure 2 the response sequence Y -> Y -> Z gives a 6 minute trip that is only one less than the optimal response sequence X -> Y -> X -> Y -> Z giving a 5 minute trip. So despite the small size of the graph, the problem is not trivial. It is easy for a RLM receiving summary evaluations to get trapped in a local optimum.

Let us now assume that the automaton does not have respondents X, Y, and Z available to it; rather, suppose it has an integer *range* of responses from 4 to 0. In other words, the automaton has a language based on integer values instead of letter symbols. If the automaton uses a partition consisting of three intervals (e.g., [4, 2] [1] [0]), then each interval represents an equivalence class where the range of responses are considered to be the same. For example, if a respondent is the range [4, 2], then the responses {4, 3, and 2} are equivalent. In general, there are 2^n ways of partitioning the range $[n, 0]$ into equivalence classes.

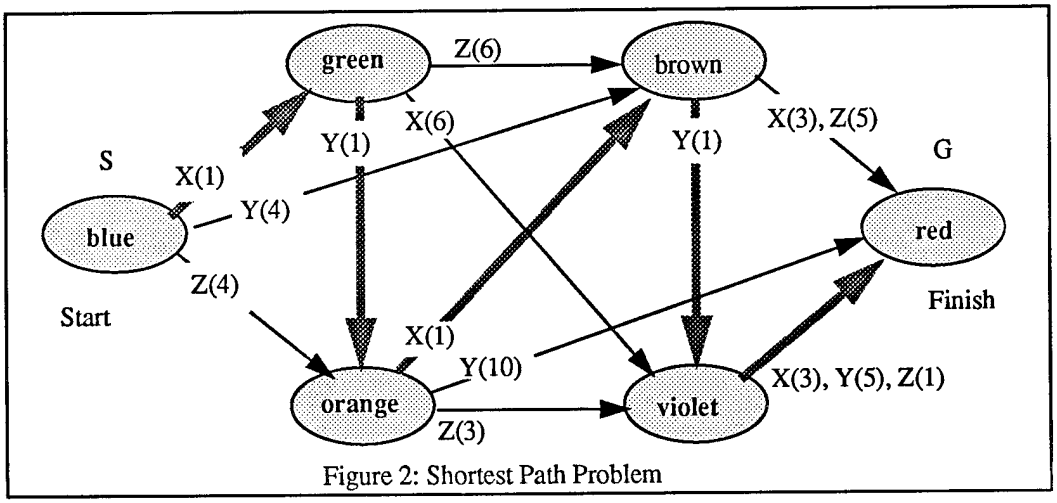

Figure 2: Shortest Path Problem

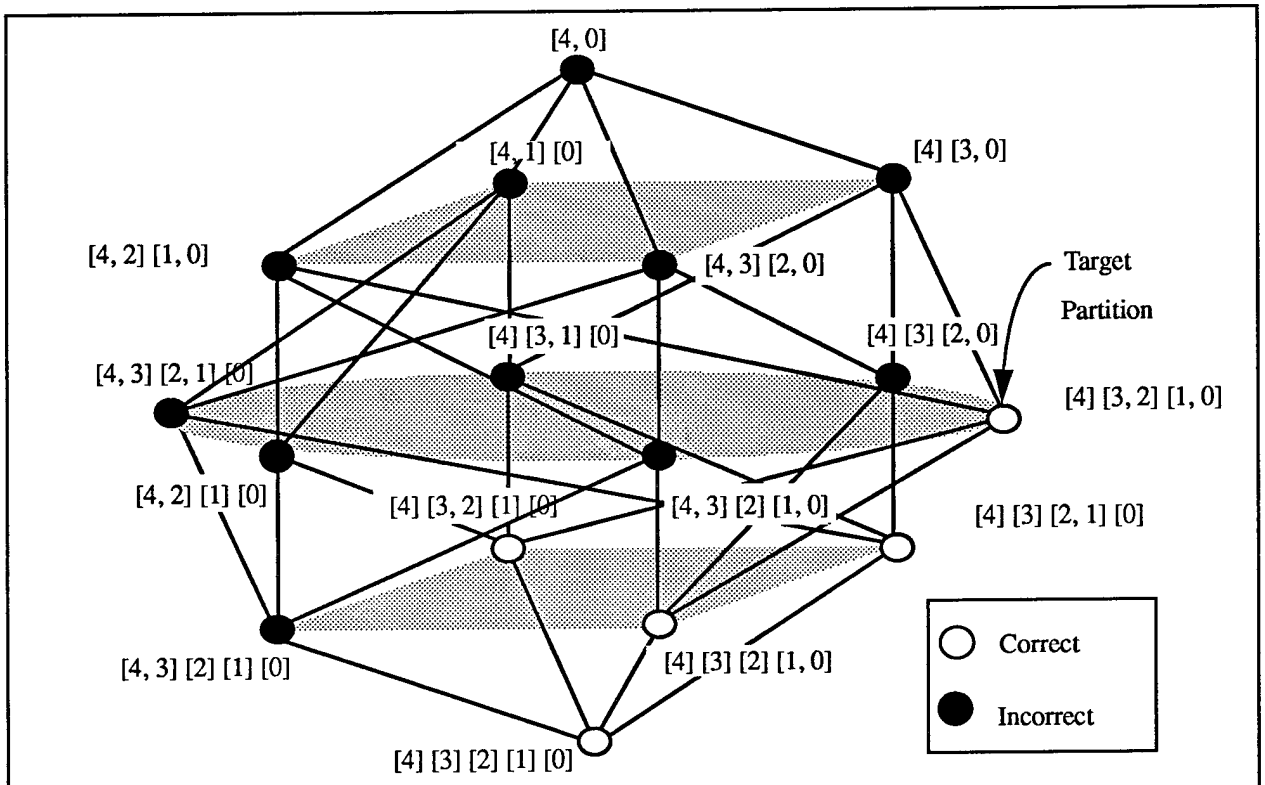

Figure 3: Lattice Represents Partitions over the Range 4 to 0

4.2 EXAMPLE OF REPRESENTATIONAL BIAS

In general, the best mapping between the environment's language and the automaton's is not known. If we consider the automaton's specific partition of the range to be an inherited characteristic, then a partition can be thought of as an intrinsic representational bias in the automaton's sensors, much in the same vein that a bat's sonar-like sensors vary in sensitivity depending on the range. This representational bias determines the structure of the automaton's STM. The partition given to the automaton remains the same during its life-span (while it solves the shortest path problem). The choice of **partition** is one of the experimental design parameters.

Figure 3 depicts the representation lattice organized with the most general representation at the top, where all responses over the range from 4 to 0 are placed in the same equivalence class (i.e., {4, 3, 2, 1, 0} = [4, 0]), to the most specific one at the bottom, where each response is considered to be unique (i.e., [4] [3] [2] [1] [0]). Thus, the strength of this representational bias can itself be framed as a search space covering the most general to the most specific representation. As we move down the lattice, each line represents an additional splitting of the range. When the environment (or the problem) maps the respondents X, Y, and Z in the graph above into one of the possible partitions, then that partition becomes the target representation, or more specifically, the **Target Partition**. For example, if X maps to [4], Y maps to [3, 2], and Z maps to [1, 0], then the target partition is [4] [3, 2] [1, 0]. An automaton that happens to inherit the target partition has the *strongest, correct* representational bias. Other representations, however, may still permit the CLA to solve the shortest path problem. Those partitions are also *correct* representations. Notice that [4] [3] [2, 1] [0] is a correct partition because there is a unambiguous mapping for each of the environment's symbols: X maps to [4], Y maps to [3], and Z maps to [0]. The range [2, 1] is not useful to the automaton because the range is ambiguous: the target partition maps Y to [2] and Z to [1]. Combining 2 and 1 together does not help the CLA, but the combination does not hurt the automaton either because there are unambiguous mappings for the symbols Y and Z. In general, the inherited partitions which allow the automaton to only converge to sub-optimal solutions, or to not converge at all, are *incorrect* representations.

4.3 EXAMPLE OF PROCEDURAL BIAS

At the end of each stage, the environment informs the automaton of the duration of its selected path through the network. The automaton's compensation function, which is defined internally to the automaton, transforms the environment's score, ξ, using an exponential function so that shorter durations receive a large reward and longer durations receive very little reward.[7] Thus the initial scaling of the reward is

$$f(\xi) = exp^{k \times \xi}, \qquad \text{(Eqn 5)}$$

where k is some small constant.

The collection entropy is also included in the compensation function. The collection entropy acts as a progressive reinforcement mechanism that effective rewards the automaton more during early learning experiences and less later on as information at decision points becomes more certain. As a result, the use of the collection entropy forces the automaton to examine more difficult-to-reach, yet unexplored, parts of the search space, even though a near-optimal solution may be easier to locate. Scott and Markovitch's DIDO system also use an entropy measure to decide what spaces to investigate (Scott and Markovitch, 1989). However, the CLA's measure incorporates a transformation of the environment's evaluation in order to maximize the reward of an experience.

Given a compensation factor, A, for adjusting the rate of change of the STM probabilities, the compensation, c, is

$$c = (A)\ (1.0 - H_c) f(\xi)\,. \qquad \text{(Eqn 6)}$$

The **compensation factor** is the procedural bias parameter modified in the experiments. All probabilities along the path (decision sequence) are increased in proportion to their current values during the update of the STM. In the experiment discussed in the results section, different levels of the compensation factor are examined in order to study the effect of applying different strengths of a procedural bias. By increasing or decreasing A, the probabilities in the STM become organized more or less quickly. These transitional probabilities in turn affect the organization of the credibility space of the hypotheses, where a trial hypothesis is considered to be a particular sequence of stimulus-response pairs. With a stronger procedural bias, the entropy of the credibility space decreases faster. However, too fast a rate of decrease can cause the procedural bias to be so strong that it is incorrect. In other words, too strong a a compensation factor may lead to a sub-optimal solution.

[7] This compensation function is similar to a reward-inaction policy, where desirable behaviors are rewarded and non-desirable behaviors receive no reward instead of being penalized.

4.4 DESIGN

In this study, we examine the 16 partitions discussed above in combination with 50 levels of compensation factor, ranging from 0.01 to 0.50. Each combination of partition and compensation factor level is repeated 20 times in order to obtain average performance values.

Two criteria must be satisfied to reach convergence: (1) the fraction of runs over the last 200 stages having the optimal solution must be greater than or equal to 0.99, and (2) the difference in the collection entropy between stages over the 200 stages is less than some small value epsilon (e.g., epsilon = 0.0005).

After achieving convergence, statistics are obtained over a window of an additional 50 stages. Two performance measurements are taken: (1) the number of stages required to reach convergence (i.e., the last stage of the window), and (2) the time-average of the exponentially transformed score $f(\xi)$ (see Eqn 5). The automaton is permitted to run up to 15, 000 stages when there is no convergence.

5 RESULTS

5.1 MEASURING THE CLA'S OVERALL PERFORMANCE (UTILITY)

Figure 4 illustrates the change in the collection entropy from stage to stage for three runs using selected levels of compensation factor: 0.05, 0.14, and 0.50. The collection entropy fluctuates as the automaton tests different trial hypotheses at each sage. Eventually, as the probability of selecting one solution becomes more certain, the entropy dramatically decreases and the fluctuations lessen. (If there were more than one solution, the change in the fluctuation would remain constant). As the level of compensation becomes larger, the number of stages required to reduce the entropy becomes smaller.

The top graph in Figure 5 summarizes the essence of the information in Figure 4. Figure 5 shows the number of stages required for the five correct partitions (i.e., [4] [3, 2] [1, 0]; [4] [3] [2, 1] [0]; [4, 3] [2] [1, 0]; [4] [3] [2] [1, 0], and [4] [3] [2] [1] [0]) for each level of compensation factor over an average of 20 runs for each combination. The target partition's line is dotted; each point has a vertical line indicating plus and minus one standard deviation about the mean. An analysis of variance and accompanying t-tests indicate that there is no significant difference among the correct representations most of the time.

The bottom graph in Figure 5 shows the convergence values of the different partitions for each level of compensation factor. These convergence values have been normalized so that the best obtainable performance is one, and the worst obtainable one is zero. The strongest, incor-

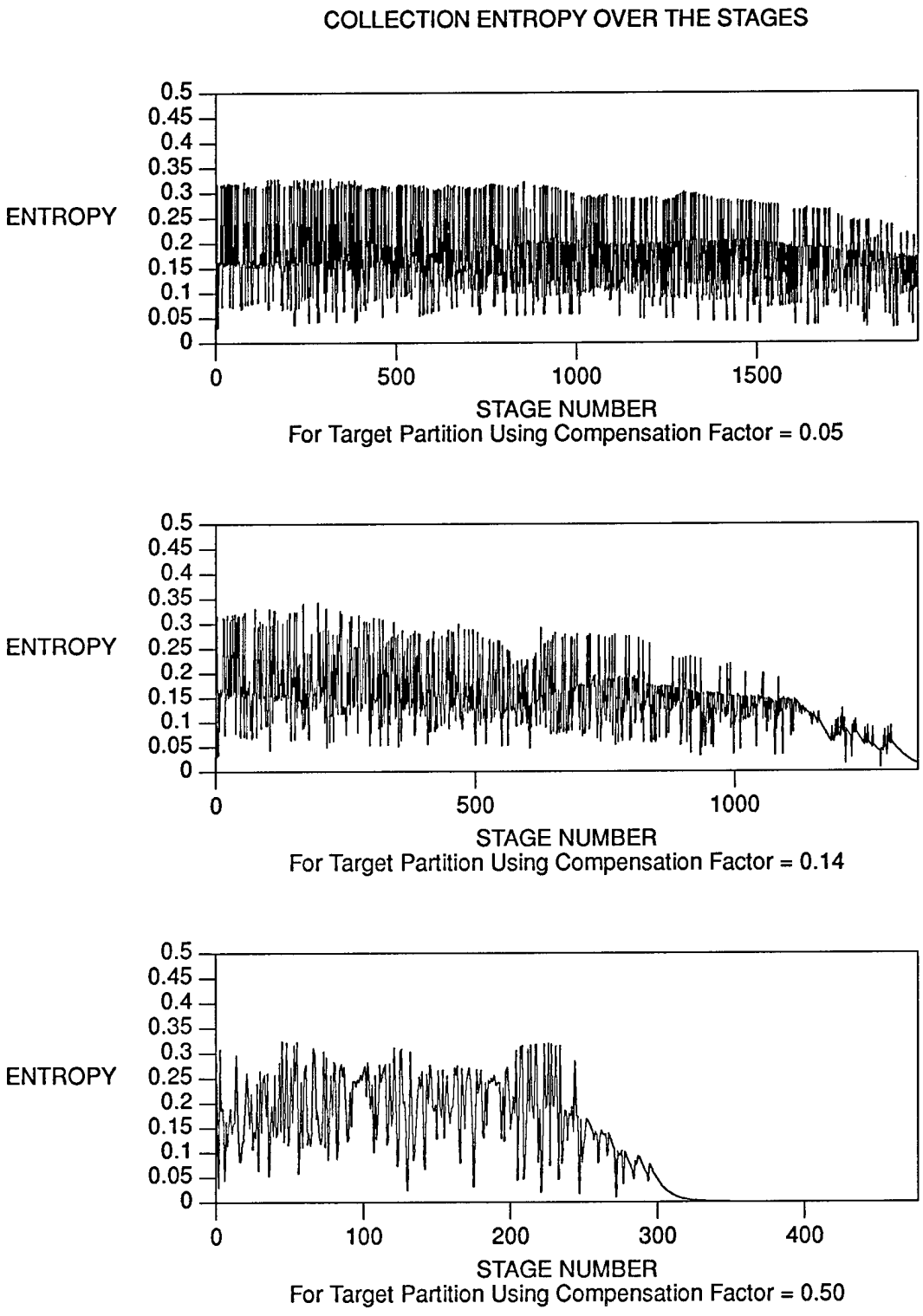

Figure 4: Example Runs Showing Change in Collection Entropy Over Stages

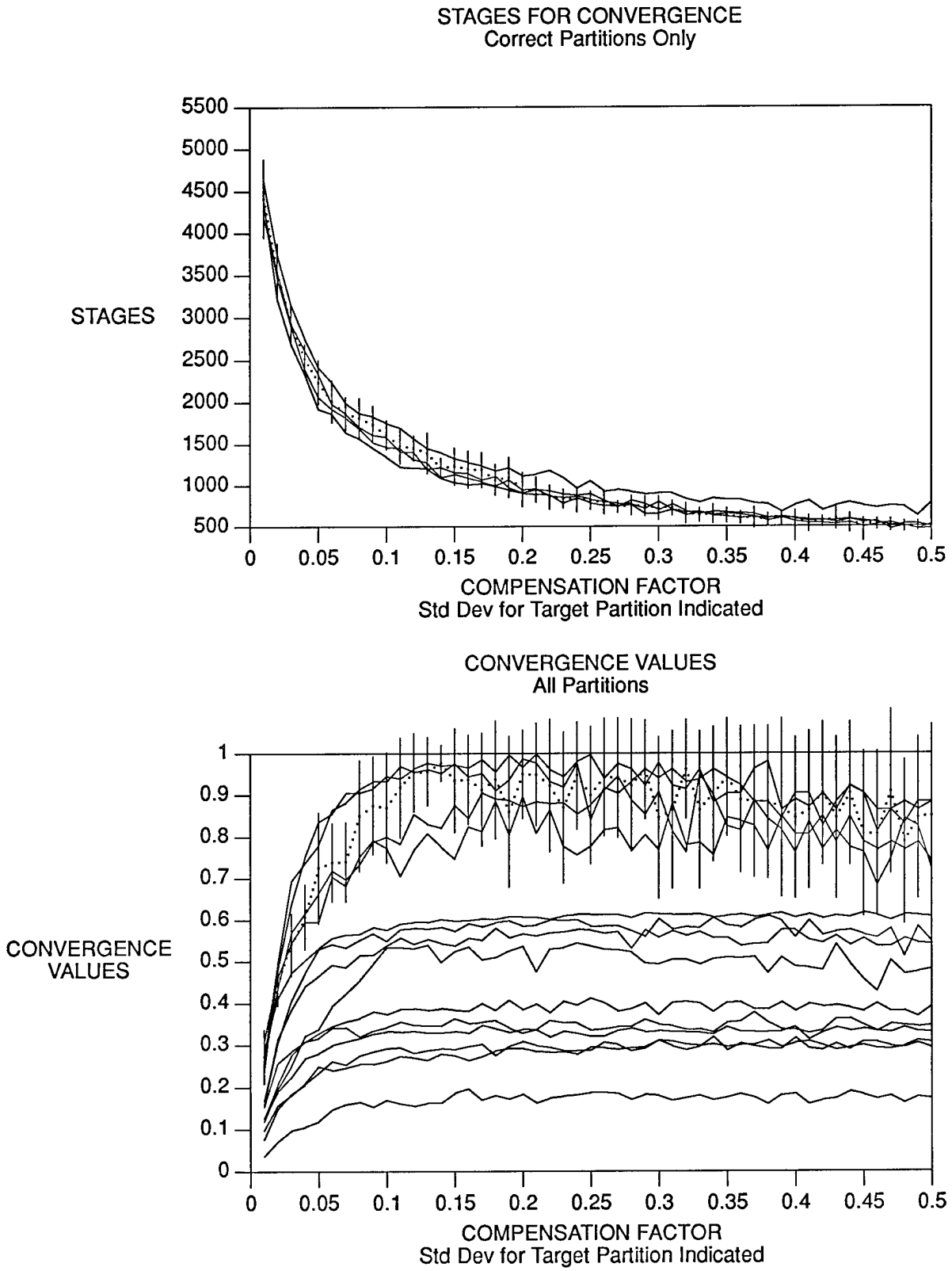

Figure 5: Stages for Convergence and Convergence Values at Different Levels of Compensation Factor

rect partition, [4, 0], lies along the compensation factor axis at a zero normalized convergence value. The top five lines correspond to the correct representations; the remaining, more spread out, lines are incorrect partitions. The dotted line indicates the target partition; with the vertical lines indicate plus and minus one standard deviation about each plotted point. An analysis of variance and accompanying t-tests indicate that there are no significant differences among the correct representations. Notice that the overall convergence values decrease for the correct partitions with a high level of compensation factor (e.g., 0.50). Even though it takes fewer stages to reach convergence (around 500), the procedural bias is so strong that some of the runs are converging suboptimally, thus bringing down the average convergence values.

6 CONCLUSION

Normally, we would expect that in the top graph of Figure 5 the correct representations would vary in their number of stages for convergence, depending on the strength of the representation. For example, the target partition, being the strongest, correct partition, should have a slightly faster rate of convergence that is significant when compared to the other correct partitions. In this problem, there is not a significant difference between the representations most of the time. On the other hand, we would not expect the convergence values in the bottom graph of Figure 5 to be significantly different for the correct partitions, since by definition of being correct, all of these partitions should allow the automaton to convergence to the optimal solution.

In the bottom graph of Figure 5, the incorrect partitions, which generally converge to sixty percent or less of the optimal convergence value, also exhibit tremendous variance in their values (this variance is not displayed in the graph). As a result, it is difficult to discriminate among the incorrect partitions, as we might expect to do, based on their normalized convergence values. Incorrect partitions rarely permit the automaton to *converge* to even a suboptimal solution. For incorrect partitions, the convergence values are primarily the result of averaging the last fifty stages over 15,000 stage runs.

In summary, correct partitions could not be discriminated on the basis of the number of stages required for convergence. Neither could incorrect partitions be discriminated based on their convergence values. One of the possible reasons for this result is that the example problem uses different symbols for the same paths through the network (i.e., word similes). The problem does not show the effect of combining different paths giving the same performance into groups. It may be that the use of similes does not degrade performance. Future research needs to address different senses of what is meant by combining terms into higher level ones.

Another problem that needs to be explored is the definition of convergence within the CLA. It may be that the current definition of convergence inherently yields results having high variance. A different definition of convergence may permit the automaton to consistently converge by same number of stages (or close to the same) when using the same combination of parameters.

This paper reviews the ideas of inductive learning biases in the context of an example RLM called a CLA. For purposes of illustration, the paper uses simple shortest path problem. In particular, the experimental work examines the performance of the automaton for various combinations of strengths in representational and procedural bias. The representational bias is the partitioning of the response range used within the STM of the CLA; the procedural bias is the compensation factor which determines the amount of increase in the probabilities within the STM. The work also introduces the use of entropy as a measure of bias strength, with particular emphasis on the strength of the procedural bias.

The work demonstrates some of the problem of discovering empirical measures of bias strength in an example RLM. Other tests cases need to be explored in order to discover when stronger representations can be ascertained empirically through performance measures.

Acknowledgments

I would like to thank Diana Gordon for her insights on inductive bias.

References

Peter Bock (1992). *The Emergence of Artificial Cognition: an Introduction to Collective Learning*, (Forthcoming), Elsevier.

Kenneth A. De Jong (1975). *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. Doctoral Dissertation, Univ. of Michigan.

Diana F. Gordon (1990). *Active Bias Adjustment for Incremental, Supervised Concept Learning*, Doctoral Dissertation, University of Maryland, University Microfilms Order No. ADG

John J. Grefenstette (1988). "Credit Assignment in Rule Discovery Systems based on Genetic Algorithms," *Machine Learning*, Vol.3, No.(2/3), pp. 225-245, (1988).

David Haussler (1987). Bias, Version Spaces and Valiant's Learning Framework. *Proceedings of the Fourth International Workshop on Machine Learning, June 22-25, University of California, Irvine*, Los Altos, CA: Morgan Kaufmann.

David Haussler (1988). Quantifying Inductive Bias: AI Learning Algorithms and Valiant's Learning Framework. *Artificial Intelligence*, 36, pp. 177-221.

John Holland (1975). *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: The University of Michigan Press.

Tom M. Mitchell (1977). Version Spaces: A Candidate Elimination Approach to Rule Learning. *Fifth International Joint Conference on Artificial Intelligence*, Vol. 1, pp. 305-310, March.

Tom M. Mitchell (1980). The Need for Biases in Learning Generalizations. *Technical Report CBM-TR-117*, Department of Computer Science, Rutgers University, New Brunswick, NJ (1980).

Kumpati Narendra and M.A.L. Thathachar. (1989). *Learning Automata: An Introduction*. Englewood, NJ: Prentice-Hall.

Larry Rendell (1986). A General Framework for Induction and a Study of Selective Induction. *Machine Learning*, Vol. 1, pp. 177-226.

David E. Rumelhart (1986). *Parallel Distributed Processing*. (Eds: David E. Rumelhart and James L McClelland). Vols 1 and 2, MIT Press.

Paul D. Scott and Shaul Markovitch (1989). Learning Novel Domains Through Curiosity and Conjecture. International Joint Conference on Artificial Intelligence, Vol. 1, pp. 669 - 674, Palo Alto, CA: Morgan Kaufmann.

C. E. Shannon (1948). A Mathematical Theory of Communication. *Bell System Technical Journal*, Vol. XXVII, July, pp. 379 - 423.

William M. Spears and Diana F. Gordon (1992). *Is Consistency Harmful?* (In the proceedings of the Bias in Inductive Learning Workshop, ML-92.)

Richard S. Sutton (1984). *Temporal Credit Assignment in Reinforcement Learning*. Dissertation, University of Massachusetts, Amherst, MA.

Leslie Valiant (1984). A Theory of the Learnable. *Communications of the ACM*, Vol. 27, No. 11, pp. 1134-1142.

Paul E. Utgoff (1986). "Shift of Bias for Inductive Concept Learning," *Machine Learning: An Artificial Intelligence Approach, Vol. II*, Chapter 5, pp. 107-148, (ed. Michalski, *et al.*), Los Altos, CA: Morgan Kaufmann.

Satosi Watanabe (1960). Information-Theoretical Aspects of Inductive and Deductive Inference. *IBM Journal*, April, pp. 208-231.

Satosi Watanabe (1971). Pattern Recognition as Information Compression. *Proceedings of the International Conference on Frontiers of Pattern Recognition*, January 18-20, Academic Press.

Satosi Watanabe (1975). Creative Learning and Propensity Automaton. IEEE Transactions on Systems, Man, and Cybernetics, Vol. SMC-5, No. 6, November, pp. 603-609.

C J. C. H. Watkins (1989). *Learning with Delayed Rewards*, Ph. D. Thesis, Cambridge University Psychology Department.

Steven D. Whitehead and Dana H. Ballard (1990). Active Perception and Reinforcement Learning. In *Machine Learning: Proceedings of the Seventh International Conference (1990)*, pp 179-188, (Eds. Porter, Bruce and Mooney, Raymond). San Mateo, CA: Morgan Kaufmann.