Standard Telecommunication Laboratories Technical Memorandum No.556, August 1966

An Introductory Description of the STeLLA Learning Machine

J.H. Andreae and B.R. Gaines

The Standard Telecommunication Laboratories Learning Automaton has been described in a previous Technical Memorandum, No. 523, from the point of view of automatic control theory (Gaines and Andreae, 1966). This account is a relatively non-technical introduction to the problem and strategies of the learning machine.

1. Introduction

A learning machine may be a satisfactory controller in a sequence of different environments by optimizing to each in turn without requiring the additional storage and sophistication necessary for it to be adequate at one time for all the environments.

Even when a learning machine is not easily programmed and fabricated in an optimal form for a particular environment, there may exist an initial form of the machine—simpler to programme and to fabricate—from which the optimal machine is trainable. Similarly, the exclusion of an optimal form by the failure of components may not prevent the machine from carrying on in a reliable manner in a sub-optimal form.

Finally, an adaptive control policy involving simultaneous decision-taking at many processing centres will give the learning machine a higher effective speed than, say, a programmed digital computer of conventional design with central processing of all decisions.

These are our main reasons for being interested in the design of a learning machine, We shall not distinguish between a "learning machine" and an "adaptive controller" except in so far as the former term implies a more sophisticated and, perhaps, less precise design.

The following formulation of a control problem for the learning machine is intentionally unspecific to immediate applications and is sufficiently general to embrace a wide range of environments, linear or non-linear, continuous or discrete, and stochastic or deterministic:

> *There is a row of lights in front of you (I-PATTERN), each light being "on" or "off" at any instant. Below the lights a row of push-buttons (ACTIONS) lie within your reach and these may be pressed, one at a time, in any sequence. There is nothing else but a single light (REWARD) below the buttons and your task is to cause this light to flash on as often as possible.*

STeLLA (the Standard Telecommunication Laboratories Learning Automaton) is a design for a storage-limited machine which will solve this problem by developing an optimal, adaptive policy for the task. The design is based upon strategies which, in turn, depend upon assumptions made about the task. Our aim has been to weaken the assumptions made, to improve the strategies incorporated and to assess the performance of the machine with a variety of tasks presented in the postulated form.

The machine and its environments have been simulated on general-purpose digital computers (chiefly on a Ferranti "Mercury" and on an English Electric " KDF 9" by courtesy of I.C.I. Ltd., and also on a Stantec Zebra). Construction of STeLLA as a mechanical tortoise was abandoned when it became clear that with present technology and available resources the building of a machine could do no more than remind us of what we had once thought interesting!

2. Strategies and structure of the machine

2.1 The Control Problem

The control problem formulated in the Introduction comprises an unknown environment with information, the I-pattern, from it, actions affecting it and the task of obtaining reward signals as often as possible. Without loss of generality the assumption is made that the information flow through the machine can be made discrete so that the machine samples the environmental information, performs an action, samples again, and so on. It is postulated that the I-pattern provides an indexing of the states of the environment which, if not complete, is at least adequate for the performance of the task. Thus the behaviour of the interacting machine and environment can be represented by a trajectory of alternate states and actions (a "path").

$$S \rightarrow A \rightarrow S \rightarrow A \rightarrow S \rightarrow A \ldots\ldots..,$$

where the short arrows imply internal decisions by the machine and the long arrows imply the reactions of the environment to the actions. At times a reward may be given and it will be supposed that this always follows an action:

$$S \rightarrow A \rightarrow S \rightarrow A \rightarrow S \rightarrow A \ldots\ldots..,$$
$$\searrow \text{(Reward)}$$

The learning machine will have solved the problem of receiving reward as often as possible when it has formed an optimal policy by attaching to every possible incoming I-pattern an appropriate action such that implementing the actions in response to the patterns attains reward with a minimum number of actions. In classical automatic control this policy is calculated by the designer using his empirical knowledge of the environment, and is then built into the machine. This built-in policy is analogous to the innate behavioural sequences shown by some animals in response to specific releasers in their environment, very useful in the appropriate circumstances but non-adaptive to changes in these circumstances. As an improvement on this one can envisage a machine which takes over the designer's task of gathering empirical knowledge about the environment and, when it has sufficient, calculates an optimal policy which it implements. However, animals seem to work in a more effective manner, forming sub-optimal policies as soon as possible and gathering data to improve them at the same time as they implement them. This approach to the optimal through adaptation of the sub-optimal is the principle on which the design of the STeLLA learning machine is founded.

2.2 The Main Strategies

The structure of the machine is derived from the four main strategies. These have been chosen to be relevant in a very wide range of environments. If they appear to be no more than "common-sense", this is but a measure of their wide validity. The division of the machine into its functional parts follows from the titles given to the main strategies, namely,

I. *Search Strategy*. In the absence of information to the contrary, random selection of actions is the best policy.

II. *Path Strategy*. The same action taken in a similar situation is likely to have a similar result.

III. *Generalization Strategy*. Two situations are less similar the less they have in common.

IV. *Prediction Strategy*. Events are more likely to recur, the more often and the more recently they have occurred in the past.

Although the main goal of SteLLA is to obtain reward as often as possible, the achievement of this goal will usually depend on the degree to which the machine can predict within its environment. Internal simulation of the way in which the environment reacts to the machine's actions is essential to the "planning" of optimal trajectories. Thus, we may state emphatically that a major sub-goal of the machine is to develop a useful internal representation in terms of the reactions of the environment. Much of the present research on the STeLLA machine is concerned with the possible interactions between the reward-seeking strategies, which cause paths to reward to be remembered, and the environment-modelling strategies which cause the behaviour of the environment to be remembered.

2.3 Search

It is through exploration or search that the machine forms its initial sub-optimal decision policies.

When a non-adaptive control procedure is given, the best search method may comprise small random deviations from this given policy, but in the absence of such prior knowledge random selection of actions is the only search procedure which will guarantee the attainment of reward, this being possible with the actions provided.

We have simulated a more complex scheme, in which environmental feedback is used, but concluded that the additional storage and decision time required would offset any advantage gained.

This procedure was based on the converse of strategy II, that a different result is more likely to be obtained if a different action is performed in a similar situation. The same idea has been extended to a "neural network" in which the "synaptic connections" between neurons are *weakened* after they have successfully conducted activity from their afferent to their efferent neurons. In this way patterns of activity in the network are made less likely to recur than patterns of activity that have not occurred, the network becoming a generator of variety of behaviour. By simulating a network of this kind on a digital computer, the variety-generating behaviour has been demonstrated in stable form and it is concluded that this could be the search mechanism of a learning machine constructed from neuron nets.

The efficiency of random search depends on the provision of powerful actions. When the actions are inadequate the cost of search alone may make the attainment of an optimal adaptive policy wholly uneconomic. In a similar way the usefulness of the I-pattern determines the efficiency of the other strategies.

2.4 Path Learning

The second main strategy is used to build up, in the machine's memory, paths to reward, the steps in these paths comprising pattern-action pairs.

When the machine is rewarded it stores the last I-pattern seen together with the action performed and labels this pair as being connected to reward. Seeing the pattern again, it recognizes it as one in its memory and performs the attached action, expecting to get reward. At the same time it stores the I-pattern it saw and the action it performed prior to recognition and labels this pair as

being connected to the pair with the recognized pattern. In this way it gradually builds up paths or sequences of pattern-action pairs leading to reward.

The pattern-action pairs form policy elements for the decision policy which operates according to the rule that if an I-pattern is recognized as being the same as a stored pattern, the action attached to the stored pattern is performed.

However, with the simple method of path formation given and with a consistent application of the decision rule, STeLLA may become trapped in a closed path or loop. Indeed it is necessary to introduce some measure of "distance from reward" and to ensure that path-following reduces this distance at each step of an acceptable path.

This is done by associating a variable weight with each connection between pattern-action pairs and with each connection between a pattern-action pair and reward. The connections are called "sequence connections" and their associated weights are called "sequence weights" The sequence weights are altered by the machine in such a way that, when a pattern in a pattern-action pair has been recognized and the action performed many times, the sequence connections from that pattern-action pair will have weights which approximate to the observed probability (i.e. frequency) with which the sequence connections represent successful steps in the machine's experience.

The most important feature of path learning in STeLLA is the way in which the success of a step is assessed.

When the I-pattern is recognized as being a pattern in a stored pattern-action pair, the "expectation" of getting to reward is calculated by treating all the sequence weights of sequence connections forming paths from the pattern-action pair to reward as probabilities that the individual steps in these paths will actually occur if the actions are performed. This expectation is the desired measure of distance from reward and, since it is derived from the sequence weights themselves, the sequence weights can only increase if their associated sequence connections consistently lead the machine to reward.

The sequence weights exhibit a compromise between the demands of the main goal, which is to be rewarded, and the sub-goal, which is to understand the environment. If storage were unlimited, it would be preferable to store two values in each sequence weight, the first value recording the probability that the transition represented by the sequence connection will occur if the prescribed action is performed when the stored pattern is recognized, and the second value recording the probability that the transition, when it occurs, is accompanied by an increase in expectation. As it is, storage is not possible to distinguish between a transition which always occurs but is seldom accompanied by an increase in expectation and a transition which seldom occurs but is always accompanied by an increase in expectation when it occurs.

It follows that the expectation is a measure of the probability that STeLLA will reach reward along paths of continuously increasing expectation. This restriction on the type of path considered ensures convergence of the decision policy represented by the tree of connected sequences of pattern-action pairs.

2.5 Generalization

The manner in which the I-pattern indexes or names the states of the environment is not known at first to the machine, but it will make a great difference to the standard of performance achieved.

On the basis of the third main strategy the machine generalizes from an I-pattern stored in a pattern-action pair to other similar patterns which are found to have an equivalent effect. This strategy enables experience in one situation to be applied to a similar situation.

What constitutes similarity is, however, a difficult question, impossible to answer in general. In the STeLLA scheme it is assumed that two states of the environment are more likely to have similar meanings and to require the same action the less they differ in the features of their I-patterns. It is assumed that each light in the row of lights, or digit in the I-pattern, represents the presence or absence of a feature of the environment according to whether the light is "on" or "off", the digit "1" or "O". The allowable difference which determines the criterion of similarity could be a fixed one built-in by the designer, but this would be a matter of computational convenience and add little to the power of the machine.

As an example of a fixed criterion, the two I-patterns

1010010111 and 0010011110

could be called similar and the two I-patterns

1010010111 and 1101001011

called unsimilar on the criterion that fewer than four corresponding digits are in opposite state in similar patterns. According to this criterion the first pair differ by 3 digits while the second pair differ by 6 digits. This criterion, like that of STeLLA, attaches no significance to the ordering of the digits.

An adaptive process is utilized whereby, if a generalization from a stored pattern to a different I-pattern is successful (that is, it leads to reward or to an increase in expectation), then the pattern digits which were neglected are made more likely to be neglected again when this stored pattern is used in the future. Conversely, if the generalization fails, neglecting of these digits is made less likely so that future use of this stored pattern will discriminate against the mis-recognized I-pattern.

This method of generalization and discrimination requires additional storage which is justified not only on the grounds that it may save more storage in terms of pattern-action pairs and sequence connections than it expends, but because it extends the range and power of the machine. The additional storage comprises weights, called "pattern digit weights" or just PDWs, associated with each of the digits of the stored patterns, one PDW to each digit.

A pattern digit weight is a measure of the importance of its digit in the stored pattern when determining the dissimilarity or "deviation" of this stored pattern from an I-pattern. The deviation of the stored pattern from the I-pattern is given by the sum of the PDWs of the digits which are in opposite states to the corresponding digits in the I-pattern. The I-pattern is then recognized as being similar to the stored pattern only if this deviation is less than a threshold parameter called the latitude. Like the expectation, the latitude is a key parameter in determining the behaviour of the STeLLA learning machine.

When the action of a stored pattern-action pair is performed as a result of the approximate similarity (i.e. non-zero deviation is less than the latitude) of the stored pattern to the current I-pattern the PDWs are altered according to whether the step is successful or not. If the step is successful, the PDWs which contributed to the deviation are decremented to make it even more likely that they will be neglected again. If the step is unsuccessful, the PDWs which contributed to the deviation are incremented and the relevant sequence weight is not decremented because the sequence connection is not likely to be required to be relevant to the I-pattern while the incremented PDWs discriminate against it.

From the above it follows that a PDW greater than the latitude prevents its associated digit from being disregarded and no I-pattern with the corresponding digit in the opposite state can be treated as similar to the stored pattern of this PDW. It is equally true that a PDW of zero value makes its associated digit to be of no consequence in the comparison of the stored pattern with an I-pattern.

Trial and error is the underlying procedure for generalization and discrimination. It raises many questions, of which these are some. When should trials be made and how often? How soon should a PDW, which is so large that it cannot be disregarded, be decreased to a value where its worth as a discriminant can be tested? Which stored pattern-action pair should be selected if more than one pattern has a deviation from the I-pattern less than the latitude? Our answers to these questions are tentative and exploratory.

All the PDWs are stored initially with high values sufficient to prevent generalization. These values are subsequently decreased with time at a rate which is slower the longer the pattern-action pair has been in useful existence. Thus, after a short probationary period trials begin and these are made more often with new experience than with old.

When more than one stored pattern is "within latitude" of the I-pattern, a weighted random choice is made giving greater weight to pattern-action pairs with higher expectation. The choice is biased towards the pattern-action pairs with higher expectation because the cost of *not choosing* these when they are appropriate will be greater and they are more likely to be correct, having been subjected to more trial and error.

2.6 Prediction

A description of how STeLLA's predictor operates is postponed until after the discussion of its purpose. This is necessary both because we do not wish to confuse the importance of its purposes with the complexity of its operation and also because the design of the current predictor leaves much to be desired.

The task of the predictor is to answer correctly, whenever asked, the question "If this pattern of lights (I-pattern) were being seen and I pushed this button (action), which pattern of lights would I see next?"

The companion question "... and would the reward light come on?" has been excluded from the predictor's task up to the present time. In more advanced forms of the machine we should expect the predictor to be concerned with reward and we have already mentioned the predictive nature of the information stored in the path memory. (The term "path memory" will be used to describe all the storage devoted to the formation of paths, the pattern-action pairs, the PDWs, the sequence connections and sequence weights). It is probably a sign of our inability to handle

complex adaptive machines that the predictor is separated from the path memory, but the resultant simplification has been helpful.

The path learning procedure and generalization cause experience leading to reward to be stored in such a way that it applies to a wide range of situations or I-patterns. Within this range of situations or "policy space" STeLLA can be said to have a decision policy for reaching reward. Outside this policy space the machine has, as yet, only the random search strategy to govern the choice of actions. It is the main purpose of the predictor to direct this search towards the policy space by enabling the machine to "look ahead". This modified form of search will be called "predictive search".

A second purpose of the predictor is to provide a check on the path-following procedure, particularly from the point of view of avoiding "foolish" errors which could be predicted from the different type of experience stored by the predictor. It is, of course, necessary for the predictor to express degrees of confidence in its predictions and to maintain measures of their reliability.

We have speculated a great deal (with the encouragement of our colleagues) on the different ways in which a predictor could be used in conjunction with the path memory to extend the potentialities of the machine. Some computer simulation has been carried out of the internal modes of the machine in which the predictor acts in place of the environment and the machine is able to explore the consequences of hypothetical and untried paths, to check and modify stored paths and, when time permits, to do what we describe in our less cautious moments as dreaming.

The predictor's ability to replace the environment promises to be of special importance when the I-pattern from the environment is subject to occasional random fluctuations or noise. If the predictor has been able to earn a high measure of reliability in such an environment, the machine may prefer the predicted I-pattern to the seen I-pattern. Such a procedure can be compared with the behaviour of a car driver on a familiar piece of road when visibility is poor.

In general, neither the simulated predictors nor the simulated environments have permitted a study of the more interesting interactions which could be induced between the path learning and the predictive parts of the machine. Research is currently focused on the design of more powerful predictors and their use.

2.7 The Simulated Predictor

The simple predictor which is used currently in the simulation of the learning machine consists of a number of arrays of adaptive elements, one array (called an "action matrix") for each action (i.e. for each push-button in the task given in the Introduction). It is the task of each action matrix to predict the next I-pattern (on-off pattern of row of lights), given the present I-pattern and the action which is about to be performed

(Given) I-pattern	+	(given) Action →	? I-pattern
(pattern-before-action)			*(pattern-after-action)*

Each "conditional" adaptive element (CE) of an action matrix has the task of estimating the frequency (i.e. probability) with which one of the digits in the pattern-after-action is a "1" when two digits (evidence) of the p-before-a are in given states (either "1" or "0"). "Unconditional" elements (UE) of an action matrix, which estimate the probabilities of states of digits in the p-after-a *regardless* of evidence from the p-before-a, are used as standard against which the

significance of the CEs is tested. If a CE estimates a probability insignificantly different from that of the appropriate UE, one of the digits used as evidence for the CE (this digit is called the "auxiliary digit") is changed in a partly random manner.

When an I-pattern and an action are given, all the elements of the appropriate action matrix are activated according to the evidence of this p-before-a and the prediction of each digit of the pattern-after-action is derived from a product of the UE probability and the ratios of the several CE probabilities to the UE probability corresponding to that digit.

The UEs and CEs of an action matrix have weights which estimate the relevant probabilities. The adjustment of these weights takes place whenever the action of that action matrix is performed, the observed p-before-a and p-after-a constituting new evidence which determines the need to increment or decrement the weights.

The predictor is an integrator of all the previous experience of the learning machine. Because storage must be limited (on account of its cost) and because the experience must be put into usable form, it is not practical to store the past experience in detail. The purpose of the auxiliary digits is to ensure that each adaptive element (CE) is performing a useful function and, thereby, to "make the most of the evidence" given limited storage. Recently we have extended the method of auxiliary digits in a way which enables the predictor to extract higher logical relationships, between the p-before-a and the p-after-a than the simple two-digit dependence achieved now.

2.8 The Overall Strategy

The overall strategy of the machine can be seen both as reaching out by prediction from where it is to where it may possibly go and also as working back through its learned paths from its desired goal to places from where it may reach that goal. The more it has to extrapolate in either direction the less certain and reliable are its strategies and, when the two extrapolations descend into unreliability before connecting, the machine has to revert to random search as the most reliable strategy.

The storage associated with prediction enables the machine to learn about its environment in general without being rewarded. The path formation and generalization procedures store goal-directed information, although this may be shared between several tasks. Much of the power of the STeLLA scheme is sought in the interaction of the environmental and the goal-directed information, but relatively little work has been done on this as yet.

3. References

Andreae, J. H. and Joyce, P. L. Brit. Patents, 1,011,685-7

Andreae, J. H. Proc. 2nd, I.F.A.C. Congress, Basle, 1963, p.497 (Butterworths 1964)

Andreae, J. H. and Gaines, B. R. STL. Tech. Memo. No. 523. "A learning machine in the context of the general control problem" (published in ***Proceedings of the 3rd Congress of the International Federation of Automatic Control (IFAC), London,*** (1966) pp.342-9. [Paper 14.B. Session 14. Tuesday 21st June 1966])