TR－H－050

Evolutionary System for the Computer Screening of the Coding Regions of Human Genome

Tetsuya MAESHIRO (JAIST Hokuriku)
Ken-nosuke WADA

1994. 2. 3

ATR人間情報通信研究所

〒619-02 京都府相楽郡精華町光台 2-2　☎07749-5-1011

ATR Human Information Processing Research Laboratories

2-2, Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02 Japan
Telephone: +81-7749-5-1011
Facsimile: +81-7749-5-1008

ATR Human Information Processing Laboratory
Department 6

Technical Report February 3, 1994

Evolutionary System for the Computer Screening of the Coding Regions of Human Genome

Tetsuya Maeshiro[1]
Ken-nosuke Wada[2]

Abstract

We present here an evolutionary system for the discrimination of the splicing sites of the human genome, whose population consists of neural networks. Most important of the parameters characterizing the networks are the learning algorithm, the duration of learning period, the number of hidden units, and the inheritance type. Three different simulations were run, and we discuss the results. Although several other simulations are necessary to be analyzed, we show that in general, even with the environment change, there is the predominance of short learning period, simple learning algorithm and Lamarckian type.

[1]Kimura Lab., School of Information Science, Japan Advanced Institute of Science and Technology, Hokuriku.
E-mail : maeshiro@jaist.ac.jp
[2]Dept.6, ATR Human Information Processing Research Lab.
E-mail : kwada@hip.atr.co.jp

1 Introduction

Through its long period of evolutionary process, life has acquired an exquisite genetic coding system. One such example is the existence of the intervening sequences, called introns, in the protein coding regions of DNA. To the proteins be synthesized, first the protein coding region must be copied to pre-mRNA, and the introns must be cut out to become matured mRNA, a process called splicing, where the introns are eliminated with the enzyme called spliceosome, and the actually coding regions, the exons, are connected together. The mRNA splicing system, consisting of some enzymes, presents an extremely high discrimination accuracy of the splicing sites. For more details, please consult [Stry88].

The details of the splicing mechanism is not yet fully known, and we used the neural networks to the discrimination of the splicing sites. Here we do not describe the implementation details of our evolutionary system. Rather, in this report we concentrate in the description of evolutionary mechanism when the individuals are free to choose their own learning algorithms, length of the learning period corresponding to the childhood, number of the hidden units of the neural network which is the power and the complexity of pattern recognition, and finally, the inheritance type, either Lamarckian or Mendelian type.

In order to analyze the general behavior of the system, the problem chosen for the simulation is a real-world problem, the splicing site problem, an attempt to introduce the real-world complexity to the system.

In section 2, we describe the learning algorithms used in our system. Total of six algorithms were used in our system. The simulation system is briefly described in section 3, and the experiment procedure and the input data in section 4. The simulation results and discussion are given in section 5, and the conclusion in section 6. For details of the simulation system, please consult [TaWa94].

2 Learning Algorithms for Neural Networks

The Back Propagation (BP) algorithm is well known as the learning algorithm for neural networks, but it has some problems like slow learning speed and high sensitivity to the initial state.

Initially, we implemented the splice point discrimination system using the simple BP algorithm, but the initial value sensitivity was too high to be used even incorporating them into an evolutionary system. Here we describe the learning algorithms used in our system.

The learning algorithms for neural networks presented here were proposed by

Ochiai and Usui [OcUs92] to improve the convolution speed. They are named Jacobs, Jacobs Hybrid, and Kick Out. Each of them are constituted by the renewal rules for weight and learning rates, and there are two renewal rules for learning rates, the $\overline{\delta} - \delta$ (DbD) rule and the $\overline{\delta} - \overline{\delta}$ (DbDb) rule, the latter one being the improved version. There are six algorithms for total, which are: Jacobs DbD, Jacobs DbDb, Jacobs Hybrid DbD, Jacobs Hybrid DbDb, Kick Out DbD, and Kick Out DbDb.

Here we briefly describe each of them.

2.1 The algorithms

First, we describe the Jacobs method. From now on, $\boldsymbol{w}$ is the weight of the neural network, $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_n)$ is the learning rate, κ is the increment factor and ϕ is the decrement factor of the learning rate, $\boldsymbol{g}$ is the gradient (first derivative of the evaluation function), and $\overline{\boldsymbol{\delta}}$ is the normalizing derivative. Then, the Jacobs method with the DbD renewal rule can be written:

Jacobs Method:

Renewal rule for weight:

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \Delta \boldsymbol{w}_k$$

$$\Delta \boldsymbol{w}_k = -diag(\boldsymbol{\eta}_k)\boldsymbol{g}_k$$

Renewal rule for learning rate (DbD):

$$\begin{array}{ll} \eta_{k,i} = \eta_{k-1,i} + \kappa & \text{if } \overline{\delta_{k-1,i}} \cdot g_{k,i} > 0 \\ \eta_{k,i} = \eta_{k-1,i} \cdot \phi & \text{if } \overline{\delta_{k-1,i}} \cdot g_{k,i} < 0 \\ \eta_{k,i} = \eta_{k,i} & \text{otherwise} \end{array}$$

where $\overline{\boldsymbol{\delta}_k} = (1-\theta) \cdot \boldsymbol{g}_k + \theta \cdot \overline{\boldsymbol{\delta}_{k-1}}$

The last formula can be rewritten as $\overline{\boldsymbol{\delta}_k} = (1-\theta)(\boldsymbol{g}_k + \theta \boldsymbol{g}_{k-1} + \theta^2 \boldsymbol{g}_{k-2} \ldots)$. This is the average gradient of the evaluation function surface on the track followed by the weight.

The DbD rule for the renewal of the learning rate is if the value is positive, it concludes that the change was small and increases the learning rate in order to accelerate the convolution. In contrary, if the value is negative, it decreases the value of learning rate concluding that it surpassed the valley where it should reach, decreasing consequently the search pace, trying to accelerate the convolution speed.

However, we cannot suppress the weight oscillation in the valley of the evaluate function caused by the renewal rule for weight. In order to solve this, the addition of the momentum term (α) was proposed, called Jacobs Hybrid method.

Jacobs Hybrid Method:

$$\Delta \boldsymbol{w}_k = -diag(\boldsymbol{\eta}_k)\boldsymbol{g}_k + \alpha\bar{\delta}\boldsymbol{w}_{k-1}$$

We treat the value of the momentum term (α) as near 1 to the acceleration effect be the maximum.

But even with the use momentum term, the weight oscillation in the valleys cannot be suppressed rapidly. Since this oscillation is caused by the fact that the fastest descent direction is orthogonal to the valley, we have to find a way to the fast descent direction be the same with the direction of the valley. The method with the compensation term to allow this was proposed, called kick out method.

Kick Out Method:

$$\Delta \boldsymbol{w}_k = -diag(\boldsymbol{\eta}_k)\boldsymbol{g}_k + \alpha\Delta \boldsymbol{w}_{k-1}$$

$$\Delta \boldsymbol{w}_k = \Delta \boldsymbol{w}_k - \frac{\boldsymbol{y}_k^T\Delta \boldsymbol{w}_k}{2\|\boldsymbol{y}_k\|^2}\boldsymbol{y}_k \quad \text{if } \boldsymbol{y}_k^T\boldsymbol{y}_k < 0$$

where $\boldsymbol{y}_k = \boldsymbol{g}_k - \boldsymbol{g}_{k-1}$

The compensation term introduced in this method serves to position the weight on the bed of valley in the case of weight oscillation.

2.2 Learning rate renewal rules

When describing the Jacobs method, we presented the DbD rule. Here we present the improved version of DbD rule, the DbDb rule.

Renewal rule for learning rate (DbDb):

$$\begin{array}{ll} \eta_{k,i} = \eta_{k-1,i} + \kappa & \text{if } \overline{\delta_{k-1,i}} \cdot \overline{\delta_{k,i}} > 0 \\ \eta_{k,i} = \eta_{k-1,i} \cdot \phi & \text{if } \overline{\delta_{k-1,i}} \cdot \overline{\delta_{k,i}} < 0 \\ \eta_{k,i} = \eta_{k,i} & \text{otherwise} \end{array}$$

The reason DbD rule cannot suppress the oscillation of weights is that on the moment just after surpassing the valley the gradient and the momentum term become unbalanced. With the DbDb rule, since the sign of the mean derivative does not change just before and after surpassing the valley, there is no decrease of the orthogonal component of the leaning rate to the valley, and the decrease occurs one step after. Because of this, the modulo of the gradient and the momentum term are balanced, correcting the weight to the direction of the valley, consequently suppressing the oscillation.

This oscillation suppressing effect of DbDb rule relies upon the value of the weight coefficient ($0 < \theta < 1$). θ should be near 1 the DbDb rule become effective.

With this rule, there is no need of learning parameter adjustment for specific problem, since we can obtain the best convolution performance.

We should emphasize here that although the weight oscillation problem was minimized with this rule, the problem of initial value sensitivity still remains, or specifically, has worsened, and hence there is a need of incorporation of evolutionary system to solve our problem, the discrimination of the splicing sites.

3 Simulator

3.1 Child and Adult - Learning Period

The neural network should learn before it can be used. In our system, the learning process is a fine tuning process, and the generation of the initial weight values is a global tuning process. Once the initial weight values are fixed, the possible values after any learning become constrained. The initial value is fixed when the individual is born, and the learning process corresponds to the childhood.

The length of learning period of the population, the neural networks in our case, could be fixed, but we made it variable in order to verify the optimal length of learning period. When the individual reaches its adulthood, it can reproduce. Since the number of the total population is fixed, an adult has to find another one and check if the opponent's fitness value is lower than his. If this is true, the loser's space is occupied with the winner's offspring. The children cannot be attacked, as if protected in a shelter.

We expect the decrease of learning times in the later stages of the simulation if there is the predominance of the Lamarckians, since if the child inherits the parent's learned data, in our case the weight and bias values, there is no need for the long learning, or even no learning at all.

3.2 Mendelism and Lamarckism

In the field of Biology, it is the common sense between the biologists that natural organisms follow the so called Mendelian evolution, where the information passed from parents to their offspring is strictly genetic, and not the sort of evolution postulated by Lamarck, who believed that natural organisms might pass acquired traits to their offspring.

This may be true for the biological world, but in the engineering field this may be different, where Lamarckians can be more advantageous than the Mendelians, since the environment change is less drastic. We believe that the Mendelian type of inheritance is better suited when the environment change is dynamic and frequent, since if the organism is hyper optimized for an environment, it may be more expensive in any sense like time, energy or some other cost measure to fit into another environment than a recent-born, who is not optimized to any specific environment.

We expect to see the predominance of Lamarckian type as the simulation goes with stable environment. In the steps immediately after the environment change, there may be a slight increase of Mendelians, since they might be more robust than Lamarckians, as latter ones are more locally optimized.

3.3 Learning Algorithms

We also let the individuals choose their learning algorithms. Ideally, they should develop their own learning algorithms, but this is unrealistic for now. Therefore, we provided six learning algorithms for total: Jacobs, Jacobs Hybrid, and Kick Out, each one with either DbD (Delta-bar Delta) or DbDb (Delta-bar Delta-bar). As described earlier, Jacobs-DbD is the simplest algorithm, which means faster, but less stable, and Kick Out-DbDb is the most complex, slower, but more stable. DbDb is slower and more complex than DbD, if compared.

4 Experiments

4.1 Data Set

The data sets used in the simulation are the data containing exon-intron DNA sequences, intron-exon DNA sequences and junk DNA sequences, the latter one containing only the exon sequences without splicing site. These sequences were picked up manually and randomly from the human genome DNA sequence data base, total of 100 sequences for each data set.

4.2 Experiments

Three simulations were run, and here we briefly describe each of them.

- Experiment 1 - The experiment 1 does not involve environment change during the simulation. It was run to analyze the general behavior of the system in a stable environment.

- Experiment 2 - The simulation starts with the positive data set containing the exon-intron splicing point. In the middle of the run, the positive data set is switched to the data set containing also the intron-exon splicing point data set. Although this is an environment change, the change is not drastic, as the switched data set contains the former data.

- Experiment 3 - As in the experiment 2, the simulation starts with the positive data set containing the exon-intron splicing point, and in the middle of the run, the positive data set is switched. However, the environment change in this experiment is more drastic than of the second experiment, as the positive data set completely changes from the exon-intron splicing point to intron-exon splicing point, the latter one being completely different from the former one.

In all experiments, the population was 50, run for 40 steps. The data set change occurred at 20th step when involved.

5 Simulation Results and Discussion

In this section we present the results of the three experiments and analyse them. Although the data set was small, the results were under our expectations.

5.1 Experiment 1

First, the experiment 1, without environment change. From the figure 1, the maximum score is 100% from the first step, since the number of data set is small, and neural networks can achieve 100% of recognition rate in one epoch with the presented learning algorithms.

We also noted that the mean score reached near 100% in 27 steps, with monotonic increase. There are some ripples of minimum value, which commonly happens in this kind of simulation.

Figure 2 shows the change of the mean value of length of learning period. It starts with around 5.5, and monotonically decreases, approximating 2 in 40 steps. This result reinforces our supposition, where in stable environment, there is no need for learning if the population is predominated with Lamarckian types.

Figure 3 shows the population composition according to the learning algorithm. At the beginning, the composition is almost equal, but as the simulation goes on, the number of those with complex algorithms decreases, reaching near 0% on the 40th step. The simple algorithms, the Jacobs, clearly predominates the population. It is interesting that the Jacobs DbDb, and not Jacobs DbD, predominates. Another interesting fact is the slight increase of Kickout DbDb between the steps 10 and 20. In this period that the Lamarckians predominates rapidly the population, as in figure 4. We interpret this as follows: The individuals with Kick Out method, which is the most powerful algorithm, learn to achieve the higher score as possible, and pass their leaned data to the individuals with simpler learning method through reproduction and mutation. It seems as if the system as whole is working together to extract the best fitted individual.

5.2 Experiment 2

Figure 5 shows the maximum, minimum and mean score of the second experiment. The environment was switched on the 20th step, and the drop of the mean score curve clearly shows that. The positive data set was not completely changed, since half of the switched data set is the same as the original one. The drop was near 15%, but the most interesting is that the maximum score was not affected by the environment change, constantly scoring 100%. Partly this is due to the small data set used in the simulation.

Figure 6 shows the mean learning times. It is interesting that the curve in the first 20 steps increases, oppositely from the figure 2, where the value tends to decrease. There is no logical explanation to this phenomena, but it may be the different random number seed used in this simulation. We cannot see any effect from the data set change, which corresponds to the environment change. The effect might have been clearer if the data set was larger. The mean value is kept almost constant even after the environment change, approximately 6.5, since the change was incremental, and not complete.

Figure 7 is the population composition according to the learning algorithm. All the six algorithms starts with the same composition, and the more complex algorithms tends to decrease. In general, as in the figure, the DbD rule predominates the DbDb rule, and Jacobs the Jacobs Hybrid, and Jacobs Hybrid the Kick Out algorithm. Just after the environment change, with a slight delay, which is approximately equal in value to the mean learning time steps, there is a slight increase of more complex algorithms, notably the Jacobs Hybrid. We believe there was no influence on the Kick Outs since there was no necessity to use the complexity and power provided by this algorithm, and the moderately powerful algorithm, the Jacobs Hybrid, was enough to manage the incremental change of the environment. Although small, there is a little increase of Kick Outs in the steps between 26 and 35. We also believe this is the environment change effect. On the 40th step, when the simulation finished, almost two-thirds of the population is Jacobs DbD, and if added with the Hybrid DbD, they account for nearly 80% of the population. Another interesting fact is the stagnation of the Jacobs DbD in the steps between 20 and 25, just after the environment change. This is due to the increase of the more complex algorithms to afford the more complex environment. After the learning is complete, the simplest algorithm begins to increase again.

Figure 8 shows the population composition according to the inheritance type. After a slight decrease of Lamarckians, they increase until reaching a stable composition, around step 15. After the environment change on the 20th step, the Lamarckians decrease a little since the Mendelians are more robust against the environment change. Note that the decrease is small, around 10%, and after the 25th step Lamarckians restart the predomination. This recovery starts 5 steps after the environment change, and 5 is approximately the mean learning time of the population at this moment, as in figure 6. It shows that the learning period of the neural networks at this moment is long enough to account for the complexity of the new data set introduced after the environment change. After the 30th step, the system keeps the stable inheritance type composition. It is interesting that the composition we got from the simulation without environment change (figure 4) is different from this simulation (figure 8). Maybe this difference is due to the different random number generator seed used in the simulations.

5.3 Experiment 3

Now we analyze the results of the third experiment. Figure 9 shows the maximum, minimum and mean score. As before, the environment was changed on the 20th step, and again we can verify the drop of the mean score curve. The mean score at the moment just before the environment change in this experiment is nearly 100%, different from the former one, where the mean score reached nearly 95%. This is due to the different seed of random number generator used. Note here that the environment change in this experiment is more drastic than that of the second experiment, since any data before the change remain after. However, we note that drop is around 15%, same as in the second experiment. This is notable, but it is partly because of the small data set used. The recovery of the mean score is faster than the second experiment.

Figure 10 shows the mean learning times. The curve until the environment change is similar to the second experiment, not showing clear decrease. However, after the environment change, there is a decrease as in the figure 2. The positive edges at around 23rd step and 27th step are due to the environment change. At 30th step, the system becomes stable, with approximately 4 learning times. It is interesting that the complete environment change allows the system to reach lower learning times than the partial environment change, as in the second experiment. Another explanation to this phenomenon is the different seeds used for random number generators.

Figure 11 shows the population composition according to the learning algorithm. A clear difference from the second experiment is the predominance of the Kick Out DbD in the middle of the simulation including the moment of environment change. It starts on the 5th step, much earlier from the environment change, which happened on the 20th step. After the environment change, the population is predominated with Jacobs DbDb, different from the second experience, where the much simpler Jacobs DbD predominated. This shows that the complete environment change demanded more power than the partial environment change. If the simulation was proceeded after 40th step, we believe the Jacobs DbD predominated the population. In this simulation, we note that the Kick Out DbD handled the environment change, passing the learned data to the Jacobs DbDb.

The clearer difference from the second experiment is shown in the figure 12, the population composition according to the inheritance type. The number of Lamarckians increases rapidly at the beginning of the simulation, achieving complete predominance before the environment change. This high inclination curve is related with the predominance of the Kick Out DbD in the middle of the simulation, with neural networks convolving rapidly because of the algorithm efficiency, consequently passing acquired data to the Lamarckians. Even with the complete environmental change, there is no effect on the predominance of the Lamarckians. This graph is completely different from figure 8, and this is due to the different seed used for random number generator. We must note that at the end of simulation, around 35th step, there is a slight decrease (10%) of the Lamarckians, and the consequent increase of Mendelians. Maybe, this composition is the system's stable composition, or maybe this is just a ripple, being the system in dynamic stability.

6 Conclusion

In this paper, we attempted to analyze the best combination of parameters when population can choose the learning algorithm, the duration of learning period, and the inheritance type. Although the simulations run are of small scale, and hence there is a need for more extensive simulations with varied parameter combinations, we could get a general outline of the behavior, where the simplest learning algorithm, shorter learning period and the Lamarckian inheritance type predominate the population, even with an environment change.

However, the quantity of simulation data is still small, and there is a need for further analysis, with larger data sets and varied simulation parameters.

This is a partial report, and we intend to collect more data related with the optimal combination of inheritance type, learning methods and the length of learning period.

Acknowledgements

The authors would like to thank Yoichi Tokura, Katsunori Shimohara, and Masayuki Kimura, and one of the authors would like to thank them for giving the opportunity to stay as trainee at ATR Human Information Processing Research Laboratory.

References

[OcUs92] OCHIAI, KEIHIRO AND USUI, SHIRO, "Improved Kick Out Algorithm with New Adaptation Rule ($\overline{\delta} - \overline{\delta}$ Rule) of Learning Rate", In: *Technical Report of IEICE*, NC92-94, 1992.

[Stry88] STRYER, LUBERT, *Biochemistry*, W.H.Freeman and Company, 1988.

[TaWa94] 田中真一, 和田健之介, 進化システムを用いた遺伝子のコーディング領域予測システムの開発, 1994.

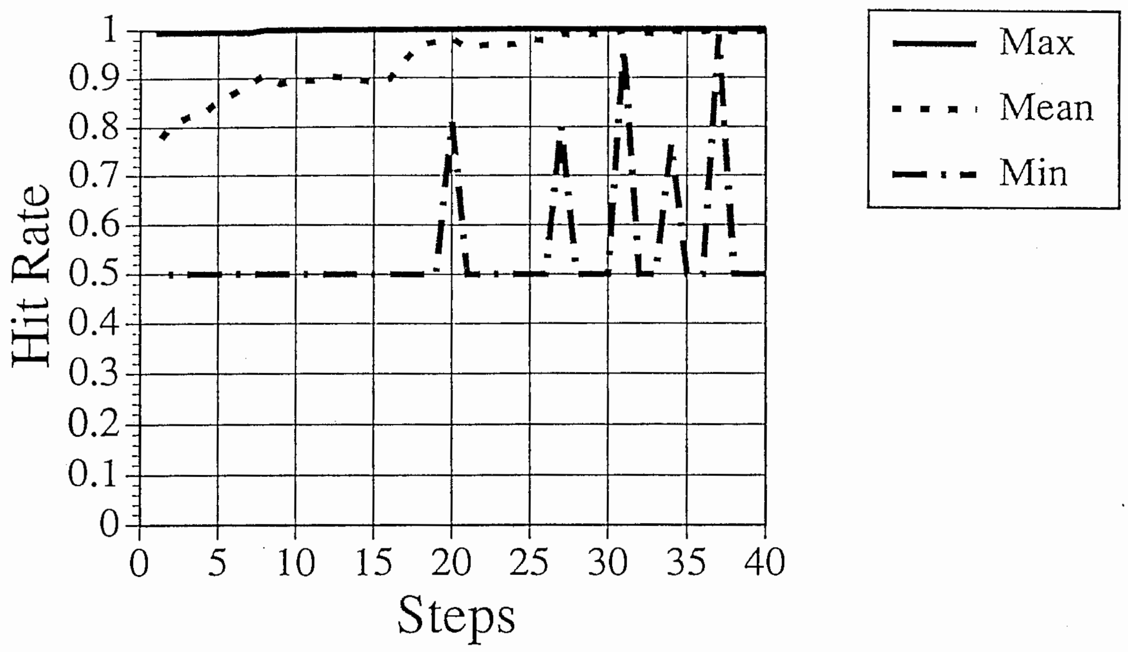

Figure 1: Stable Environment: Max, Min and Mean Scores

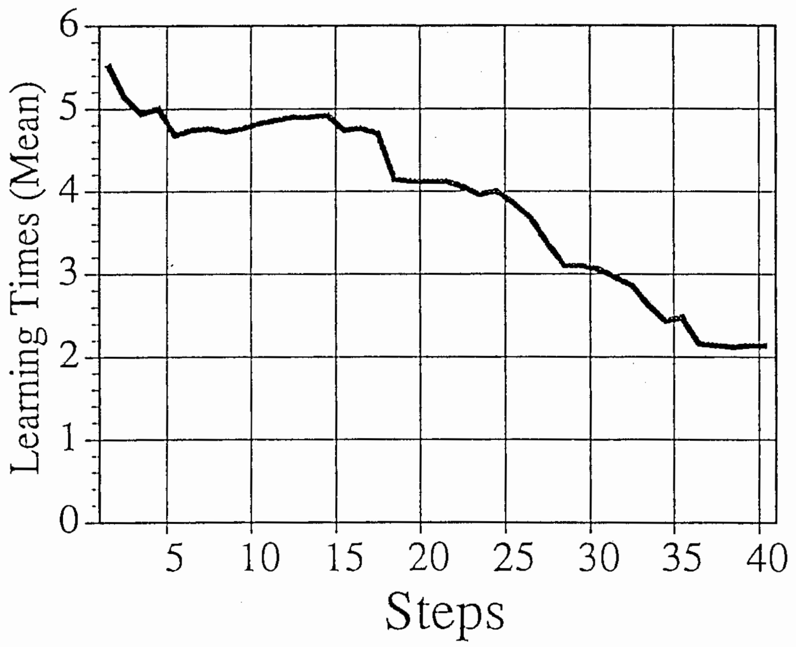

Figure 2: Stable Environment: Learning Times

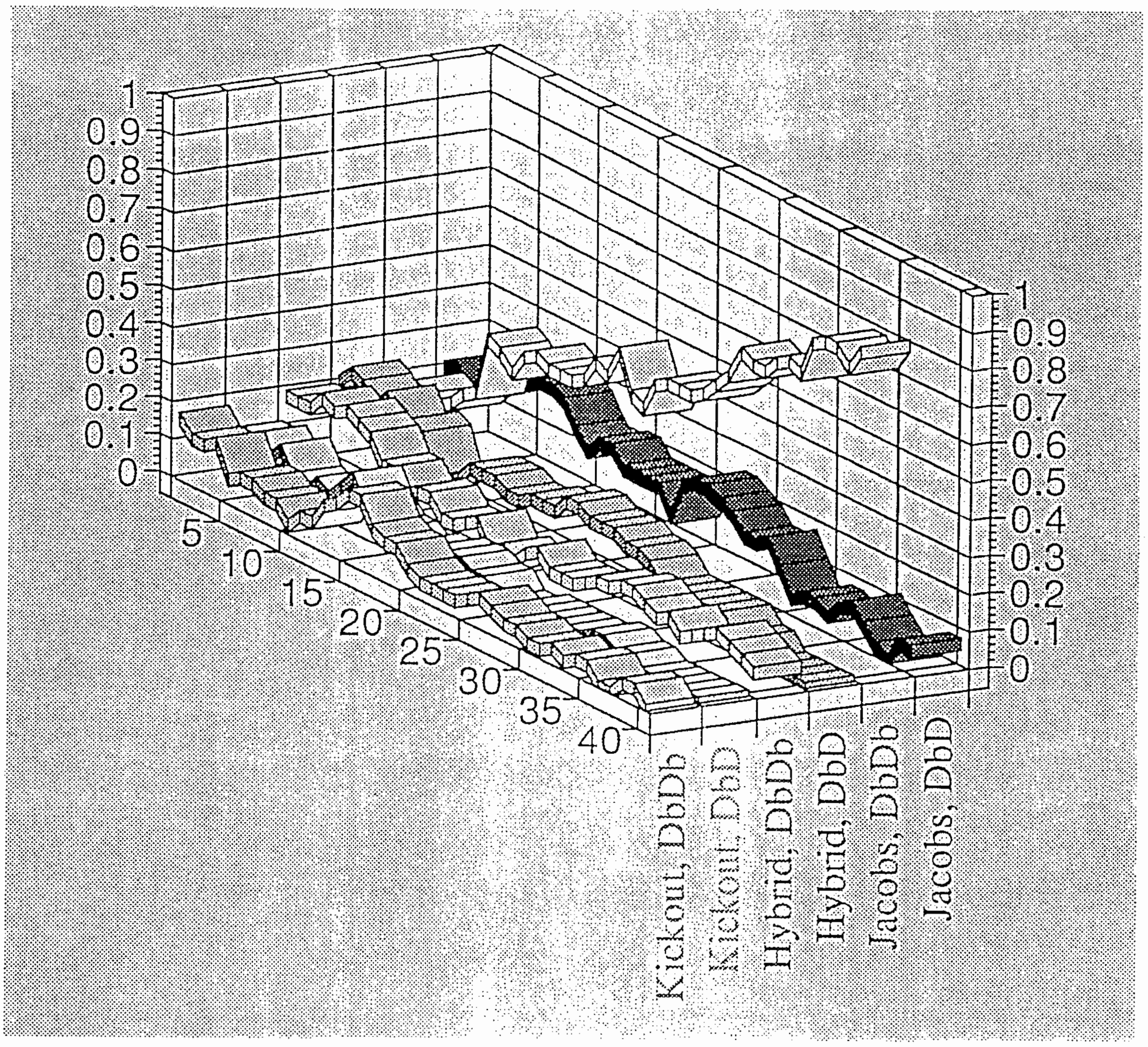

Figure 3: Stable Environment: Population Composition by Learning Method

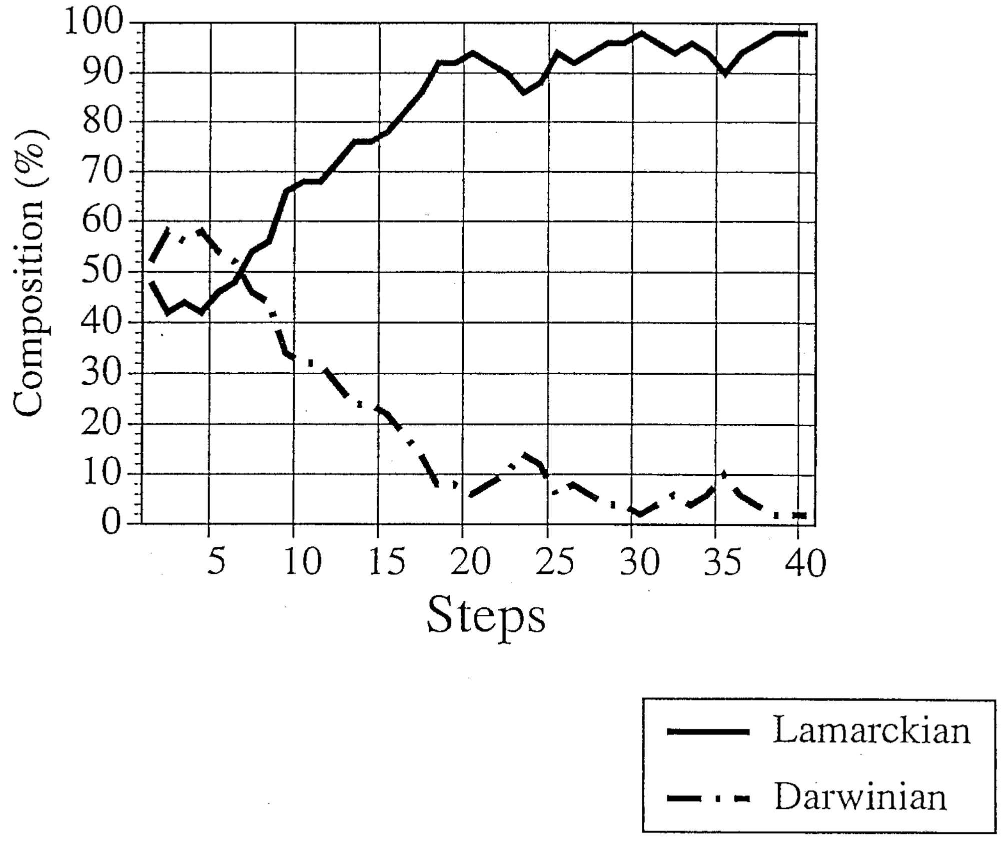

Figure 4: Stable Environment: Population Composition by Inheritance Type

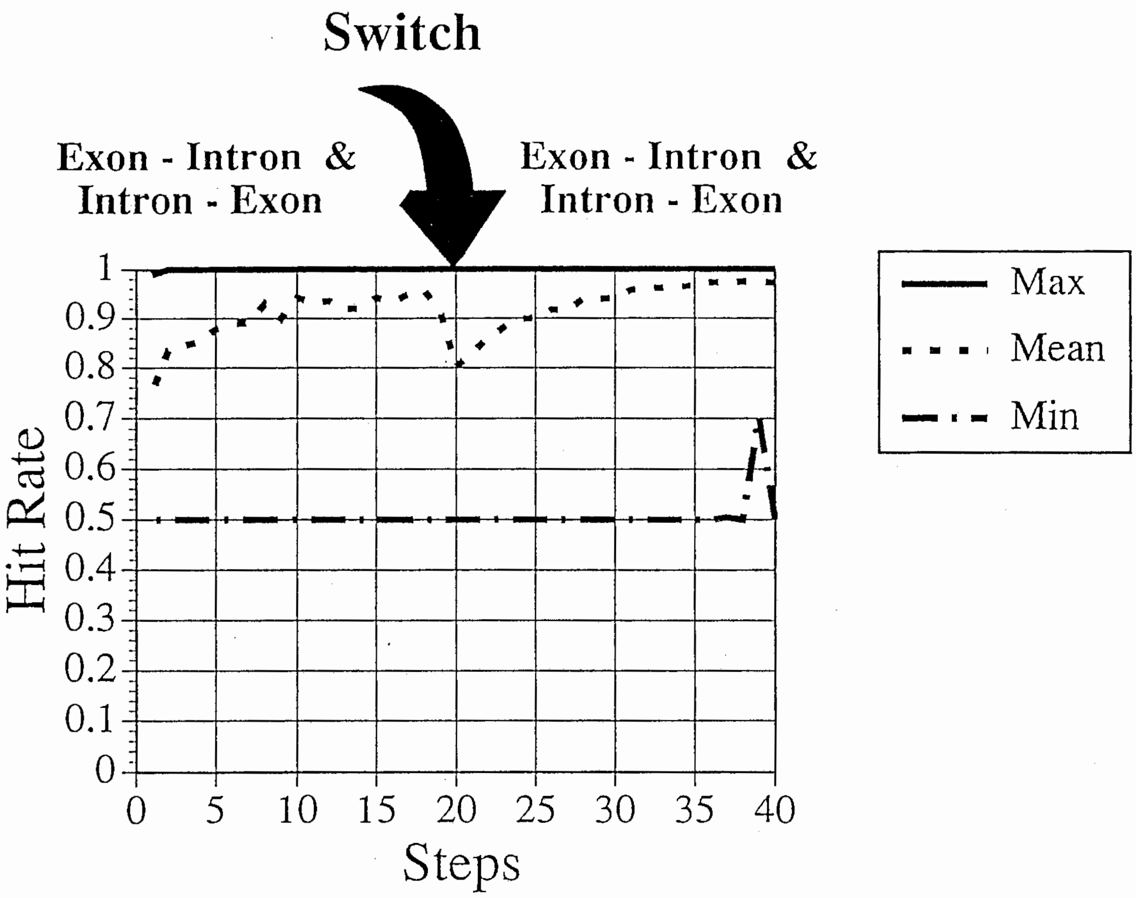

Figure 5: Experiment 2: Max, Min and Mean Scores

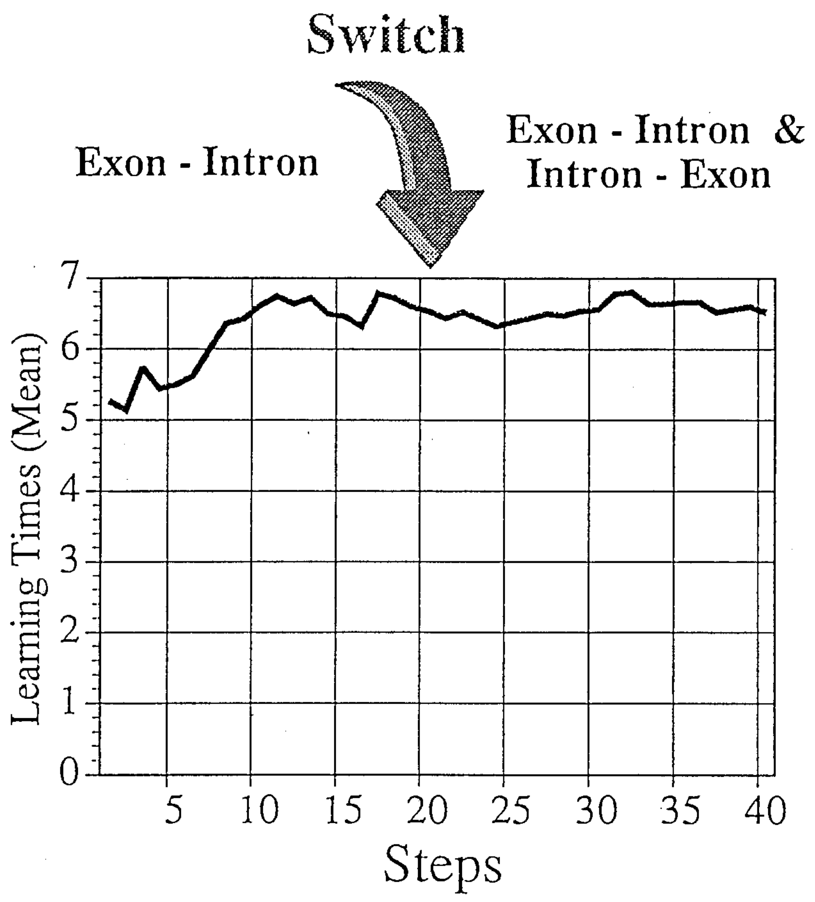

Figure 6: Experiment 2: Learning Times

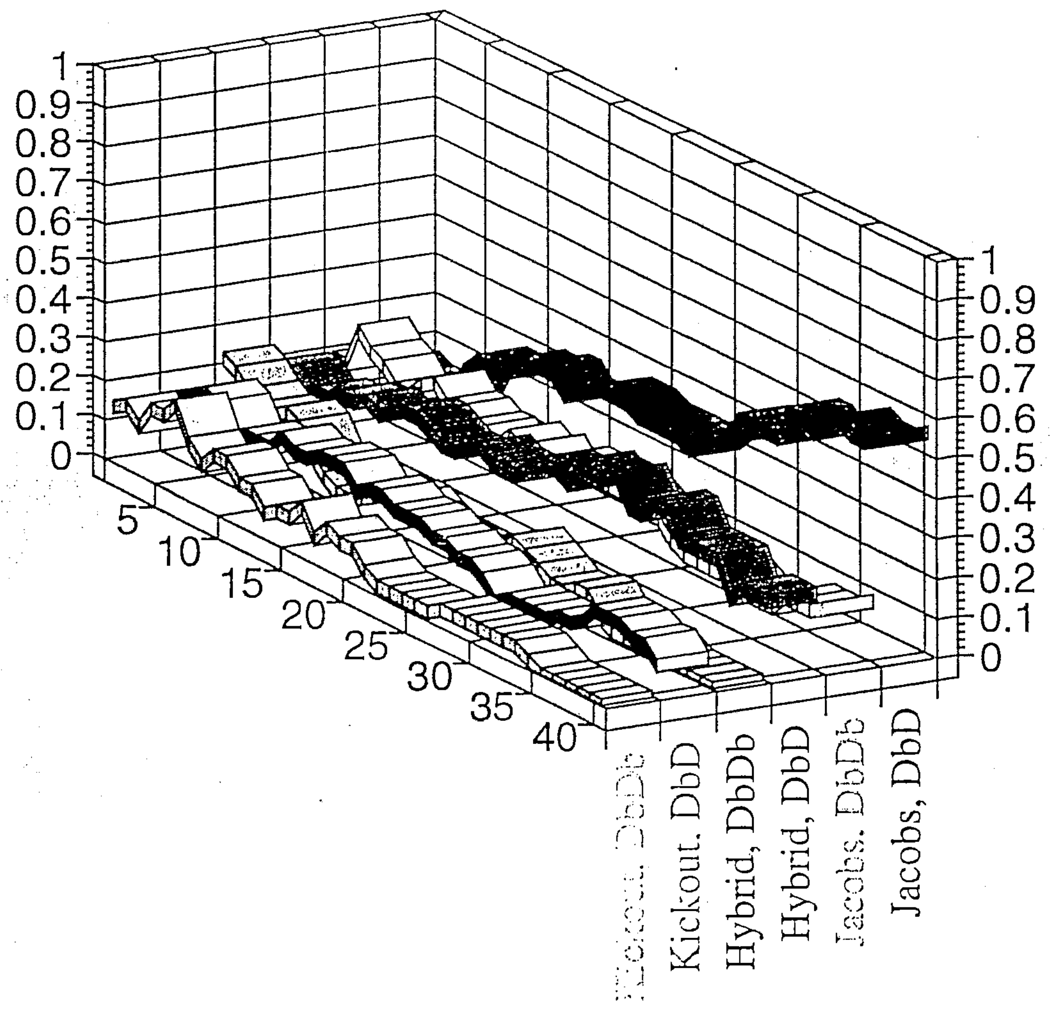

Figure 7: Experiment 2: Population Composition by Learning Method

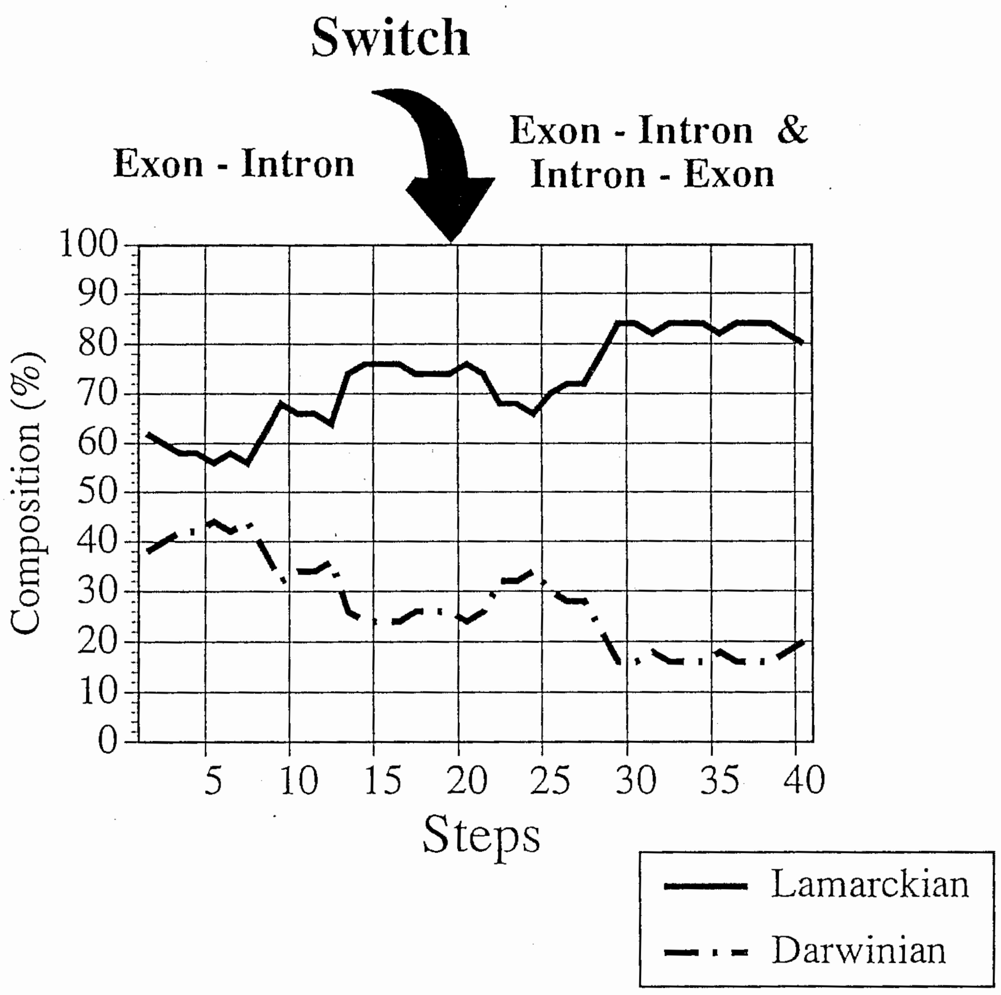

Figure 8: Experiment 2: Population Composition by Inheritance Type

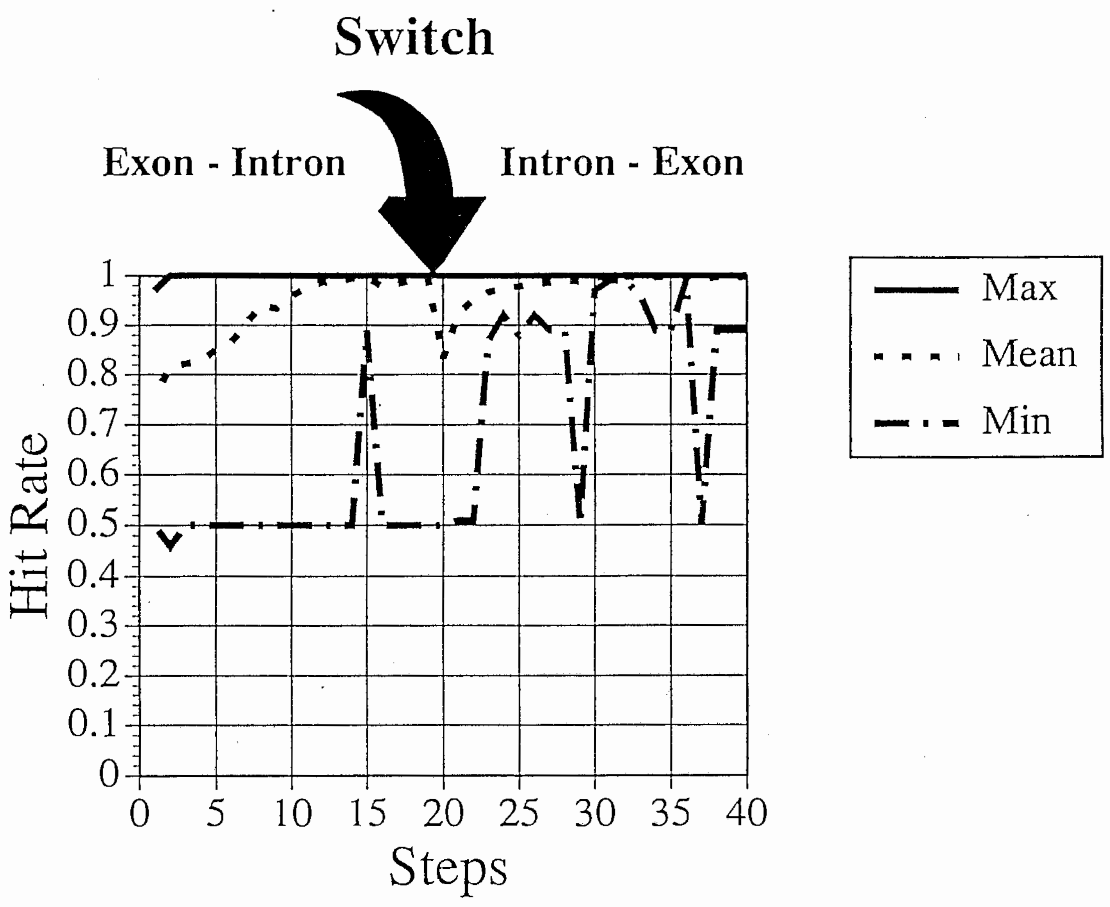

Figure 9: Experiment 3: Max, Min and Mean Scores

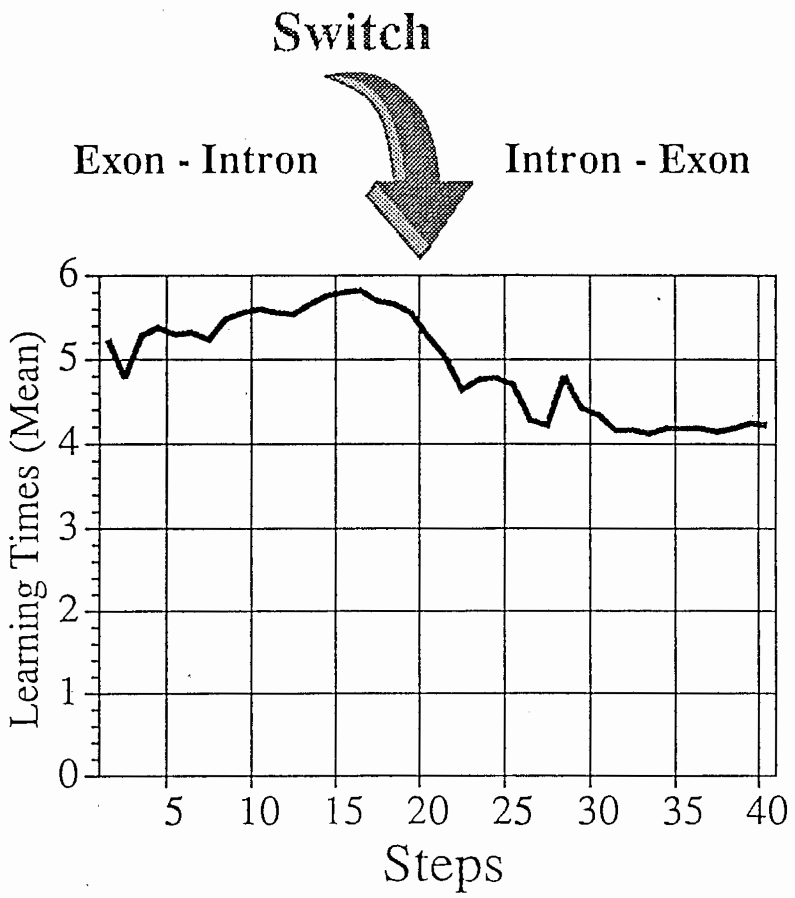

Figure 10: Experiment 3: Learning Times

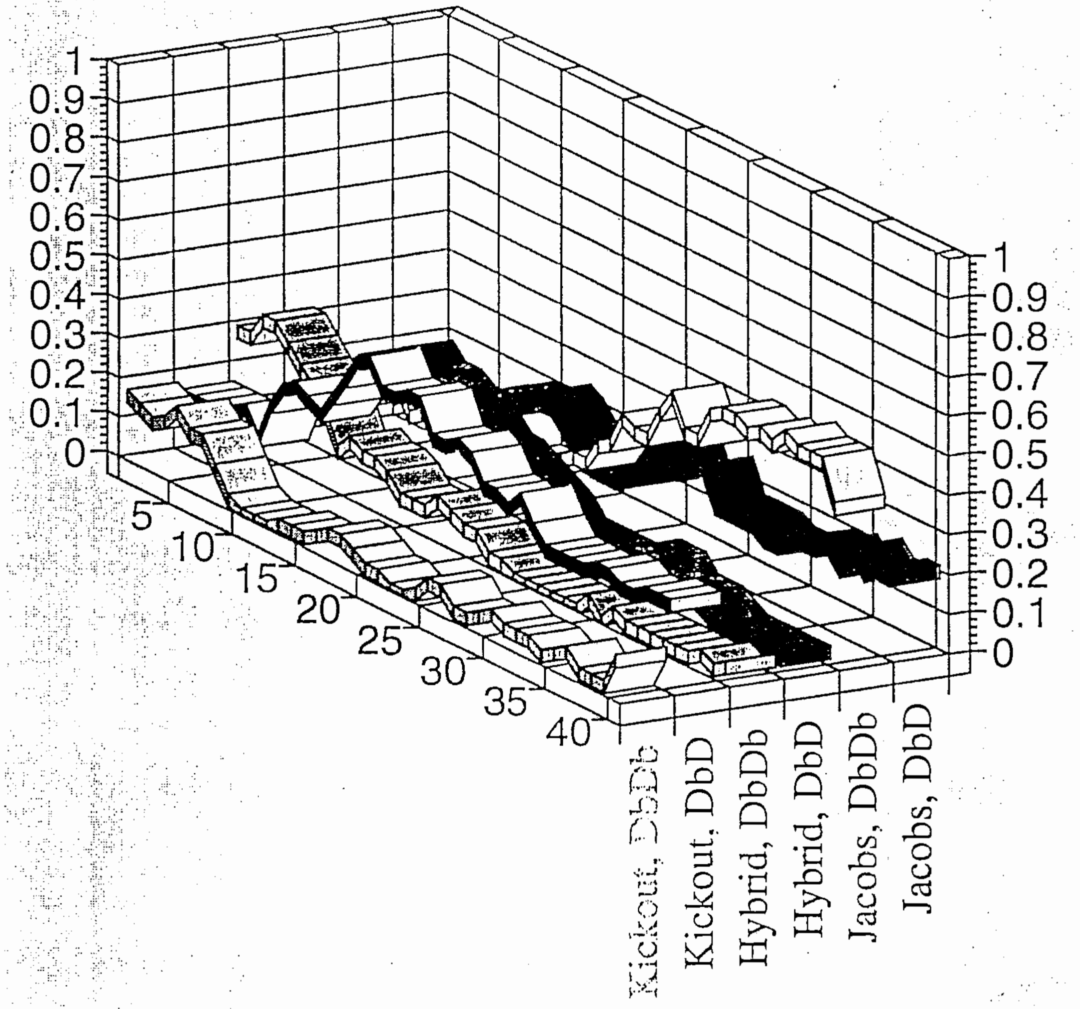

Figure 11: Experiment 3: Population Composition by Learning Method

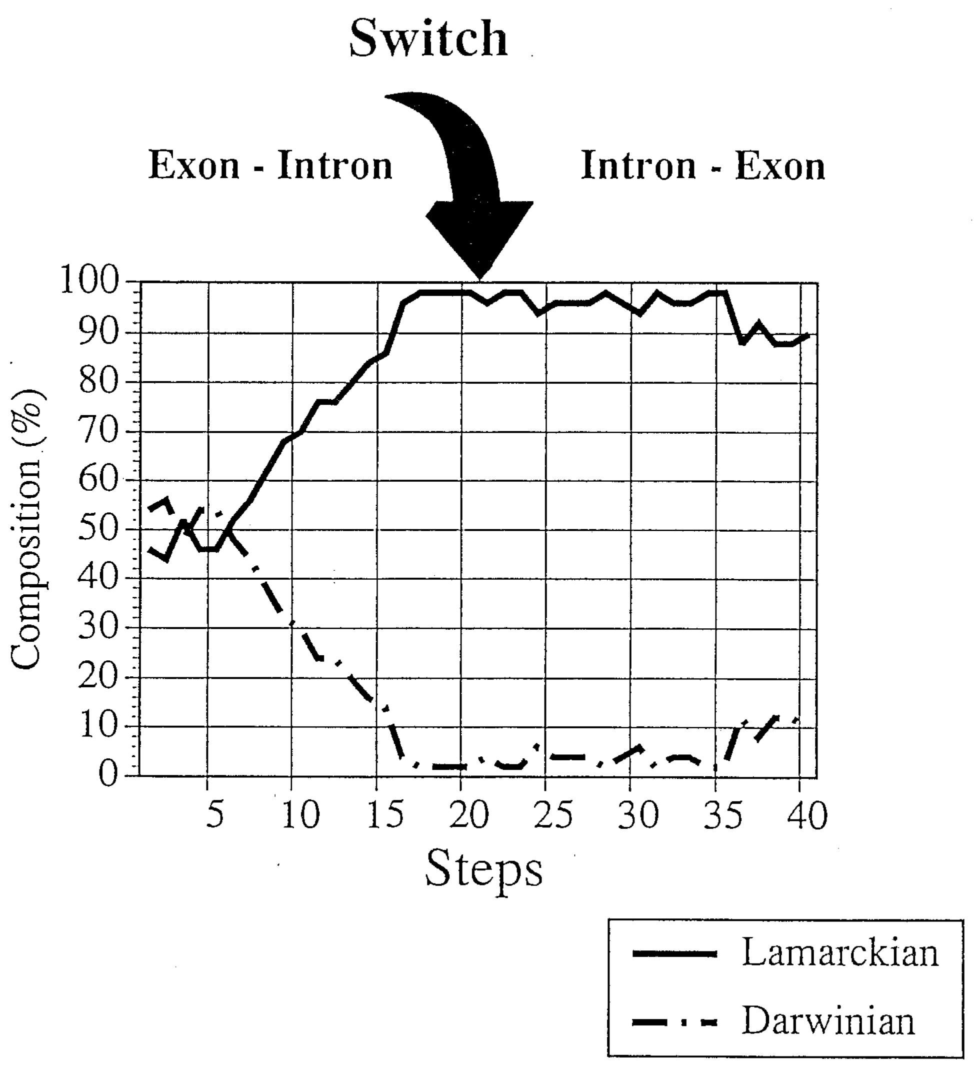

Figure 12: Experiment 3: Population Composition by Inheritance Type