

# Layered Learning for Evolving Goal Scoring Behavior in Soccer Players

Andrei Bajurnow  
School of Computer Science and  
Information Technology  
RMIT University  
Melbourne, Australia  
abajurno@cs.rmit.edu.au

Vic Ciesielski  
School of Computer Science and  
Information Technology  
RMIT University  
Melbourne, Australia  
abajurno@cs.rmit.edu.au

**Abstract**—Layered learning allows decomposition of the stages of learning in a problem domain and can have many positive effects, both on the amount of computation required for learning and for program comprehension and verification. We apply this technique to the evolution of goal scoring behavior in soccer players and show that layered learning is on average able to find solutions comparable to standard genetic programs more reliably and in a shorter number of evaluations. The solutions evolved with layers have a higher accuracy but do not make as many goal attempts. We compared three variations of layered learning and find that maintaining the population between layers as the encapsulated learnt layer is introduced was found to be the most computationally efficient and led to solutions of high quality with greater reliability than the other layer integration methods. Despite these advantages, the quality of solutions found by layered learning did not exceed those of standard genetic programming in terms of goal scoring ability. Complications arose when the first layer learnt a skill utilising a terminal which took an extended range of values in the learning of the second layer. Also, layered learning in this fashion requires a large amount of domain specific knowledge and programmer effort to engineer an appropriate layer and the effort is not justified for a problem of this scale.

## I. INTRODUCTION

Layered learning is a bottom up machine learning technique allowing a decomposition of tasks or abilities to be learnt in a particular problem domain. This permits a sequential break down of the learning that must occur and allows leverage off the skills and abilities previously developed to accelerate the rate of learning. In this sense, a layer is a specific skill to be learnt. The combination of layered learning and genetic programming has the potential to be highly beneficial, because the complexity of the tasks to be learnt within the layers are greatly simplified compared to the problem as a whole. This simplification allows improvements in the number of evaluations taken to evolve a solution, the quality of the abilities evolved and the scalability of learning.

When learning with a layered approach, questions arise as to the effectiveness of the learning within a layer and what is the best method for integrating it into further evolution. We have chosen to employ layered learning in the evolution of goal scoring ability in soccer players using a very simple function set to evaluate the approach in this domain.

In attempting to score goals the player must learn to find the ball on the field from any possible initialisation and then learn to kick the ball into the goals while avoiding a static goal keeper. This scenario lends itself to a decomposition of two layers. The skill to be learnt in the first layer is learning a kick vector to avoid the goal keeper. The skill for the second layer is the ball finding ability. In the first layer, a percentage of the total learning recourses are dedicated to learning goal kicking ability where the player and ball are initialised within striking distance from the goals and the goal keeper must be avoided. Following this the player and ball are randomly initialised on the field and the player must evolve the ability to find the ball and then execute the goal kicking ability previously learnt.

### A. Aims

This research aims to assess the benefits of genetic programming with layered learning applied to evolving goal scoring behavior in soccer. Evaluating this will require an analysis of the quality of behavior evolved and the amount of computation required to achieve it in genetic evolutionary terms. Two research questions are posed relating to the properties of the learning that occurs:

- 1) Does layered learning evolve higher quality solutions than standard genetic programming for goal scoring behavior and what is the best method for integrating it into future learning: layer encapsulation with population reinitialisation, layer encapsulation without population reinitialisation or incremental learning without the encapsulation of the layer?
- 2) Are there improvements in computational efficiency and scalability using layered genetic programming compared to standard genetic programming for evolving goal scoring behavior?

## II. GENETIC PROGRAMMING APPLIED TO SOCCER

Previous work in genetic programming and Robosoccer [1] has shown that it is possible to create soccer players which can score goals in a game of soccer. Matkovic [2] demonstrated the advantages of co-evolution in an attacker versus defender situation. Luke [3] fielded a team of eleven in the Robocup 97 competition, which made it to the third

round of the RoboCup tournament before being eliminated by the eventual winner. Keepaway soccer [4][5], while not being a full game of soccer, demonstrates successful cooperative behavior. Ciesielski et al [6] investigated the effects of high level primitives compared to low level primitives and the relative levels of soccer playing ability achieved.

### III. LAYERED LEARNING APPLIED TO SOCCER

Layered learning requires the partitioning of a problem into separate skills to be learnt in each layer. Once a skill has been learnt, the next layer can start learning and is able to utilise the skills developed in all the previous layers. All this learning is situated in an environment, in this case a soccer game. The environment must be carefully manipulated to provide the right conditions for the required skill to be learnt within a layer. An appropriate reward function must also be crafted appropriate to the environment to ensure that desirable behavior is learnt.

Layered learning is a machine independent and problem independent learning scheme and as such there is some variation in what is considered to constitute a layer. Applied to soccer, Stone[7] defines three distinct layers at differing abstraction levels within a multi-agent soccer domain. The first is the individual skill of intercepting the ball learnt by training a neural network. The second is ball passing skills and evaluation learnt by a decision tree, which requires the ability of ball interception. The third layer was pass selection by reinforcement learning, utilising both the previous skills. Each of these layers are encapsulated and do not change during further learning. When applied to the evolutionary algorithm, layered learning permits guiding of the search by allowing the fitness function to rate differing abilities at different stages. Gustafson and Hsu[4][8] applies layered learning in genetic programming to Keepaway soccer, a game where a team of four players must prevent a single opposition player for coming into contact with the ball. Two layers are used, the first with the goal of maximising the number of accurate passes and the second being to minimise the number of ball turnovers to the opposition player. These do not involve any new player abilities (kick, turn and move being the only functionality present). The technique used is that of switching the fitness function to the second fitness function after a set number of generations and continuing evolution with the same population. The layer is considered to be the level of playing ability present in the population as the fitness function is switched. It is hoped that the abilities evolved can be retained and permeate through the population while the new ability in the second fitness function is being evolved. While this is a decomposition of separate skills which feed into future learning, there is no concrete code structure that can be pointed to as being the layer. This approach is similar to incremental learning in genetic programming [9] in that it does not contain concrete structures or specific modules to represent the layer.

### IV. IMPLEMENTED SYSTEM

#### A. SimpleSoccer Simulator

Evolutionary learning using the RoboSoccer simulator is very slow as games must be played in real time or near real

time. This means that it can take several days to perform a few thousand evaluations on a single processor. Results in other evolutionary learning investigations, for example ice hockey [10], suggest that hundreds of thousands of evaluations are necessary. Thus we have used a much simpler soccer simulator, which supports the same general concepts as RoboCup but provides much faster running of games.

The simulator used for these experiments [11] is based on the AsciiSoccer simulator [12], which is a further simplification of the RoboSoccer simulation league. SimpleSoccer is a simulator environment for testing soccer behavior. It uses floating point coordinates which resolve into cells on the field. Soccer players have a 45 degree vision cone with a limited range in the direction they are currently facing. They can face in any direction from 0.0 to 360.0 degrees. They have the ability to move and turn in any direction. For each player's turn in the game loop, its evaluation function is called. This function generates the action that the player will perform by executing the evolved genetic program. The action is calculated based on the current field state. There is no messaging between player and server and each player has direct access to the state of the field when computing its next move. The environment does not include stamina, momentum or the randomness present in RoboSoccer. This reduces the simulation processing required and allows for easier analysis of behavior. It is hoped that behaviors learnt on this simpler level might become the foundations for behaviors in the full RoboSoccer server.

#### B. Layers Defined

The experiments performed here are aimed at developing the highest level of goal scoring behavior against a static goal keeper in the most computationally efficient manner. A natural decomposition of the problem domain for evolving goal scoring ability against a static goal keeper is goal kicking (with the aim of avoiding the goal keeper) and ball finding ability (in order to kick more goals). Rather than evolving both skills simultaneously as would occur in standard genetic programming, it is hoped that some efficiency will be gained by allowing the evolution of goal kicking in its own layer.

#### C. Stationary Goal Keeper

With the goal keeper present, it is not sufficient to repeatedly execute `kick(TheirGoalDir)` followed by `dash(BallDir)` which serves the purposes of both dribbling and scoring. The stationary goal keeper was simulated by canceling any goals scored when the ball entered the center of the goals. `TheirGoalDir` returns a vector to the centre of the goal so by canceling any goals scored along this vector, the problem difficulty was increased. An offset from `TheirGoalDir` now had to be learnt and can be partitioned as a layer.

#### D. First Layer Defined

The first layer is intended for learning a goal scoring kick to avoid the goal keeper. A single player can dedicate resources (population generations) to learning how to kick a goal without

Layer One:



Fig. 1. Layer initialisation example

the need for developing the ability to find the ball as required in the full scenario. To facilitate this, the initialisation of the player and ball on the field is engineered to make it easy to evolve this skill. The initialisation for this layer (see Figure 1) has the goal kicking player (represented as '<') facing the goal and positioned next to the ball (represented as 'o'). The player can be initialised randomly anywhere along an axis five cells away from the goal it is attacking. This provides a variety of vectors to the goal for the player to learn an offset to kick in order to avoid the goal keeper. The scenario should accommodate the various angles that the player might have to kick from during a game with enough variation that the behavior will generalise to any angle of attack. The first layer evolves for seven hundred generations from these possible field initialisations. This ensures that extremely high ability will be learnt every run. The fitness is goal only fitness and in this scenario, the optimal solution is to make one kick at the goals and avoid the goal keeper.

The technical method used for making what was learnt in the layer available to future generations is *encapsulation* [13] of the best program found in the first layer. At each generation, the highest scoring player is saved. At the completion of generation seven hundred, the fittest program is inserted into the terminal set for the next layer to use for the generations thereafter.

### E. Second Layer Defined

Once goal kicking ability has been learnt to an acceptable level in the first layer, the required abilities are expanded to include searching for the ball and maneuvering it into position before attempting a goal kick. The key issue here is whether the continuing evolution can incorporate the first layer to accelerate the learning.

The second layer is the full scenario of random initialisation of the player and ball on the field, such that ball finding and maneuvering ability must be learnt before the skill of goal kicking evolved in the first layer can be executed. The presence of the encapsulated goal kicking layer should reduce the amount of learning required to efficiently score goals.

### F. Run Parameters

TABLE I  
RUN PARAMETERS

elitism	1%
mutation	29%
crossover	70%
max tree depth	5
population size	250
generations	10 000
total fitness evaluations	2 500 000
number of games per evaluation	1

The genetic parameters and number of evaluations for the standard genetic programming runs and the layered runs were kept constant to facilitate comparison between the learning achieved in identical environments (Table 1).

The experiments were performed on Xeon 2.8 GHz cpus. Each node has between one and two gigabytes of RAM per node. Each evolutionary run was submitted to a single node and ran for approximately seventy-two hours of CPU time.

### G. Function and Terminal Set Defined

A function and terminal set (Table II and Table III) was established that allows a single player to score goals in a soccer game. The only actions the player can execute are kick, turn and dash. Also included are angle mathematics, direction mathematics and representations for the goals of both sides. Ball direction and distance are included so the player can make calculations in relation to the ball's position on the field. These ball specific functions will only return a value if the player has the ball in sight else null will be returned.

TABLE II  
FUNCTION SET AVAILABLE TO GENETIC PROGRAMS. NOTE: BASIC MATHS FUNCTIONS OMITTED

Function Name	Return Value	Result
if(bool, action1, action2)	Action	if bool true, action1 else action2
kick(angle, distance)	Action	Player kick
turn(angle)	Action	turn to indicated angle
move(angle)	Action	move in desired direction
dmath(distance, distance)	distance	distance math operands
amath(angle, angle)	angle	angle math operands

TABLE III  
TERMINAL SET AVAILABLE TO GENETIC PROGRAMS

Terminal Name	Return Value	Result
OwnGoalDist	distance	returns distance to own goal
OwnGoalDir	angle	returns angle to own goal
TheirGoalDist	distance	returns distance to opponents goal
TheirGoalDir	angle	returns angle to opponents goal
ballDir	angle	direction of ball
ballDist	distance	distance to ball
ballisvisible	boolean	true if player can see the ball
hasball	boolean	true if the player can kick the ball
Angle	angle	initialises to a random angle
Distance	distance	initialises to any random distance

```

if (goals kicked >= 1)
  then fitness =
    number goals * 100
if (own goals kicked >= 1)
  then fitness =
    fitness - number own goals * -100
if (fitness < 0)
  then fitness = 0

```

Fig. 2. Fitness Function

#### H. Fitness Function

The fitness function used rewarded goal scoring only (Figure 2). This makes learning difficult as there is no information to guide the search until a player accidentally kicks a goal. Until this occurs, the search is effectively random.

#### I. Test Suite

For every fitness test of a player, the initialisation on the field is random. In order to determine a measure of the best player evolved, a sample of the top players is made and each plays one hundred pre-generated games. The suite of one hundred test games were created by randomly generating one-hundred field initialisations and saving them for reuse in every player test.

### V. SOLUTION QUALITY AND LAYER INTEGRATION

#### A. Aim

To determine whether layered learning evolves higher quality solutions than standard genetic programming for goal scoring behavior and what is the best method for integrating the learnt layer into the continuing evolution. The methods compared are layer encapsulation with population reinitialisation, layer encapsulation without population reinitialisation or incremental learning without the encapsulation of the layer.

#### B. Experimental Design

Thirty runs of each layered method are generated for comparison with standard genetic programming. The number of goals scored and relative accuracy achieved in the one-hundred game test suite are calculated and compared.

- Standard genetic programming will be used as the point of comparison. This does not include a layer and all required skills must be learnt in the full scenario with no domain decomposition.
- Layered learning with encapsulation and population reinitialisation has a percentage of generational learning dedicated to learning goal kicking skills. After the number of generations dedicated to learning in the layer, the best program is encapsulated and introduced into the terminal set for continued evolution. The population is deleted and a new one generated randomly to ensure an even distribution of the new layer node in the continued learning.
- Layered learning with encapsulation and maintaining of population has the same layer but the population is

retained and not deleted. The new node can only be introduced by mutation.

- Incremental learning without layer encapsulation has the same layer but at the end of the allotted generations, the best program is not encapsulated and the population is retained for the continued evolution. The population at the end of the first layer act as a seed for the continuing evolution. The evolution must evolve ball finding behavior without losing the ability to score goals efficiently. The player and ball initialisation on the field is the only change between the layers.

#### C. Results

Goal scoring behavior of varying degrees of ability was evolved by both the standard genetic programming runs and the layered learning runs. Comparisons can be made along two dimensions. The first is that of cumulative goals scored and the second is of goal kicking accuracy, the aim of the learning in the first layer.

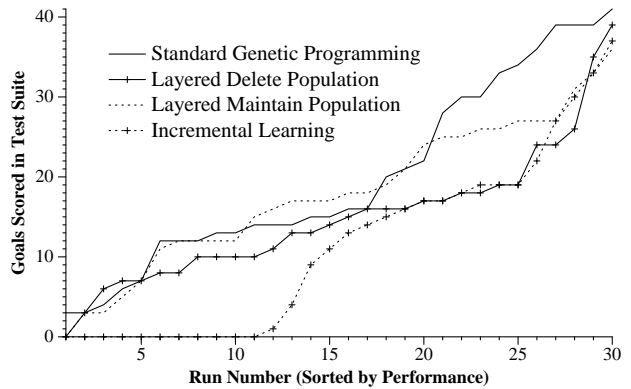


Fig. 3. Comparison of goal scoring ability of four techniques. Goals scored in each run by best players in each run.

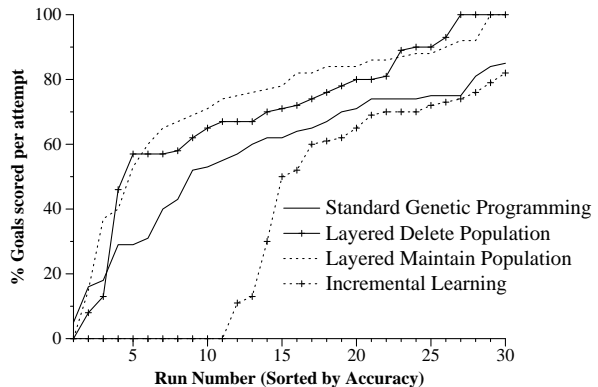


Fig. 4. Goal kicking accuracy of four techniques compared.

Figure 3 compares the goal scoring ability of the best players found in each of the thirty runs as ranked in the

test suite. Each line represents one of the techniques trialed and plots, in ascending order, the maximum goals scored by each of these best individuals. The runs are sorted in order of increasing goal scoring ability to demonstrate the distribution of fitness achieved. The standard genetic programming runs are included to facilitate ease of comparison with previous discussions. Figure 4 compares the accuracy achieved in goal kicking and each run is sorted as in the previous figure.

Any significant difference in playing ability based on goals scored is not supported. A t-test at 95% confidence levels of average goals scored by standard genetic programming compared to the encapsulated layered learning methods fails to show a difference in raw goal scoring ability (the purpose of the evolution). While this is evidence against suggestions that the layered approach is better, it does demonstrate that it is capable of producing behavior equivalent to standard genetic programming.

It is immediately evident that an incremental evolution approach is the least reliable for this kind of layered approach. While the best player produced is capable of scoring goals with equivalent frequency to other techniques, thirty percent of the runs do not evolve the ability to score goals. It is suggested [9] that incremental evolution is suited to learning subsets of increasing complexity in the problem domain. The layer used here is not so much a simpler version of the full game but rather a separate skill entirely.

Figure 3 also shows that using layered learning with encapsulation and maintaining the population from the first layer performed very reliably with only one run in thirty not scoring a goal. An analysis of the goal kicking accuracy (Figure 4) shows this method is the most reliable in evolving players with high accuracy. This evidence is supported as statistically significant with a t-test at 95% confidence levels showing that the learning of the angle offset to avoid the static goal keeper was learnt better by the layered approach. It appears that the persistence of the ability to kick accurately in the form of the encapsulated layer allows the population to quickly escape from the converged state of the population after the switch.

#### D. Analysis

Table IV summarises the performance of the techniques. Total goals are the total number of goals scored by the best players from each of the thirty runs of each technique as evaluated in the test scenarios. Accuracy is the percentage of successful goal kicks out of all the attempted goal kicks (poor accuracy means that most goal kicks went straight to the goal keeper). Effort is a measure of how much learning was required to score the goals counted in the cumulative goals column. The figure represents the number of evaluations spent per goal.

The most reliable technique to integrate the layer into the continuing evolution is layer encapsulation while maintaining the population into the next learning environment. Given the same computation recurses, this develops the highest accuracy and scores the highest number of goals with the lowest effort.

## VI. COMPUTATIONAL EFFICIENCY AND SCALABILITY

### A. Aim

To determine if there are improvements in computational efficiency and scalability using layered genetic programming compared to standard genetic programming for evolving goal scoring behavior.

### B. Experiments

The data for this analysis is from the experiments performed in the previous section.

### C. Results

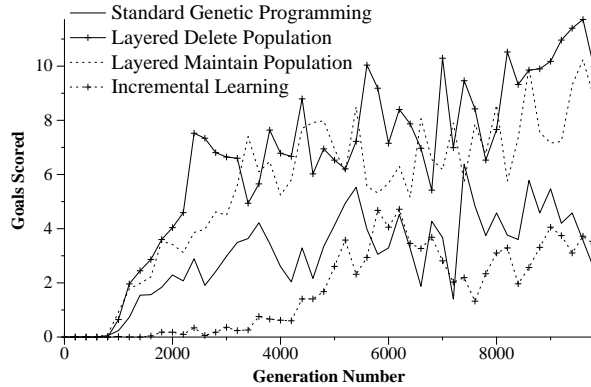


Fig. 5. Comparison of learning rate averages (30 Runs) for all methods. Note: The first seven hundred generations of the first layer are not represented.

A comparison of the learning rates, measured by the number of goals being scored by the fittest player at a particular generation during evolutionary learning, averaged over thirty runs (Figure 5), showed the high variability of the learning achieved in the runs. The graph displayed is a moving average fifty generations in length. The first seven hundred generations are omitted for comparison purposes. The following discussions will refer to this graph.

The first thousand generations of standard genetic programming had few goals scored by any of the runs. Evolutionary learning seems to overtake random search after generation 1 000 where the plot begins to rise. The rate at which competent, reliable goal scoring behavior is learnt varied so widely that the graph oscillates between a fitness of two and six which was quite sporadic over the thirty runs.

The rate of learning compared to standard genetic programming was accelerated in both the techniques which employed encapsulation in creating the layer. Approximately double the number of goals were being scored given the same computation time by these methods. The learning differential between the techniques was maintained over the full 10 000 generations. At generation 10 000 the average for encapsulated layers with population reinitialisation was around thirty, indicating that nine goals were scored at this point across the thirty runs. This is a factor of 2.5 greater than the standard genetic programming runs. This acceleration of learning is supported

TABLE IV

LAYER TECHNIQUES COMPARED (BEST PLAYERS SELECTED FROM THE 30 SEPARATE RUNS OF EACH METHOD AS EVALUATED IN THE TEST SUITE)

Method	Total goals	Accuracy	Effort
Standard Genetic Programming	601	57%	124792
Encapsulated Layer with Population Re-initialise	449	68%	167037
Encapsulated Layer retain Population	563	74%	133214
Incremental Learning	341	56%	219941

by the fact that the average generation where the fittest players were found for layered learning is 5 000 and generation 7 000 for standard genetic programming.

#### D. Analysis

The programs generally started by kicking in `TheirGoalDir` or at some constant angle. This would allow a portion of the kicks to score goals, provided the player was initialised with an accomodating angle to the goals (Note: at an acute angle kicking with `TheirGoalDir`, the ball will resolve from floating point positions into a cell at the side of the goals, thereby avoiding the static goal keeper). This is suboptimal behavior as it is not making a calculation based on the player in relation to the goals and the required offset. The next learning stage had some players learning a function which used `BallDir` in some way. As the ball was always initialised at 180 degrees from the player in this layer, this will work (see Figure 1 in section IV). The optimal solution is a simple offset addition to the `TheirGoalDir` angle, which allows any kick to enter the side of the goals from most angles. Examples found in the best layered players are  $(a+ \text{TheirGoalDir } a351)$  or  $(a+ a7 \text{ TheirGoalDir})$ .

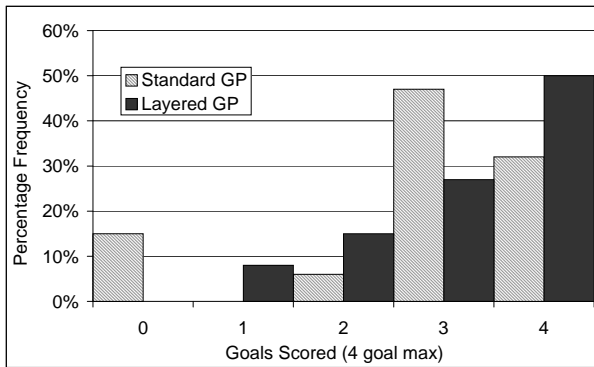


Fig. 6. Frequency of the highest fitness achieved during evolutionary learning over thirty runs. The maximum fitness in a test game is four goals.

A cumulative frequency chart (Figure 6) of the highest fitness found in each evolutionary run (not the test suite) reveals that layered learning did out perform standard genetic programming in terms of reaching the highest fitness of kicking four goals in fifty percent of the runs. A definite skew to the higher fitness range was also evident with no runs in the

zero fitness bin. This is contrasted with the standard genetic programming runs in which fifteen percent of the runs had no goals scored and the highest percentage of evolved playing ability is found to be three goals, rather than the maximum four goals. From this perspective layered learning offers greater reliability in evolving goal scoring behavior.

The apparent advantage of layers in accelerating the population learning rate averages did not seem to translate directly into a higher proportion of fitter players. The first reason is that kicking accuracy (what is learnt in the first layer) is only one component of soccer playing. The player still must find the ball on the field and maintain sight of it while dribbling towards the goal. If this ability is not evolved, then the player is at a considerable disadvantage and no degree of accuracy in goal kicking can compensate for this.

The second reason is that if a layer is not an optimal solution, it will hinder the player rather than advance its fitness. In some cases a player would run to the ball only to kick it out of bounds in seemingly random vectors. Closer analysis revealed that the kick layer contained `BallDir` which was a constant in the first layer (due to the fact that the ball was always initialised at an angle of 180 degrees from the player), but became a variable in the full game due to random initialisation. This a serious defect and is a major hazard in a program tree encapsulation algorithm such as the one used here. Once encapsulated, there is no possibility for further enhancement or refinements of behavior.

These issues might help explain the apparent contradiction between the layered methods scoring more goals on average in the learning runs but not scoring higher in the test scenario. The ability to kick accurately that was learnt in the first layer might allow a goal to be scored in a particular field initialisation, which might occur with some frequency such that over the thirty runs, at least one player will encounter it in each generation, therefore boosting the score slightly. When the test scenario is run on the best players, the field initialisations are well distributed over the whole field so that any slight advantage is reduced.

On the graph (Figure 5), the learning differential seems very large. To put it into perspective, the highest average score achieved by standard genetic programming at any particular generation was six goals in total by all the best players found at this generation in the learning. The highest average score achieved by the layered methods was twelve goals being scored across the thirty generations at this point. This is not displaying a significant difference in goal scoring ability. Only ten percent of the total possible goals are being scored. If the score was closer to one hundred and twenty, it would mean that

at that across all the thirty runs, the best performing players of each run in that generation would be scoring the maximum four goals possible in each training game. The playing ability never comes close to this level of ability and in this light, the differential does not seem so wide.

By introducing a new node into the function and terminal set of the full scenario, a greater search space is created. The new node, however, summarises concisely a key element required for sufficiency and is therefore more efficient. This alters the relationship between the nodes when switching layers and the new population in the full scenario must evolve an integration with the behavior node from the first layer. This integration can be seen in the way that the kicking ability encapsulated in the first layer node is favored over any direct call to `kick()` in the program trees in the full scenario.

It is not guaranteed, however, that the evolutionary process will integrate the behavior in future learning. There was one instance of a high scoring player from the layered learning runs which did not use the learnt node. After reinitialising the population, the ability to score goals was relearned by the evolutionary process without reference to the layer node. This is always a possibility in a random process and is clearly a large waste of computational recourses.

Learning in genetic programming is more efficient with small trees due to the reduction of the search space. For a large problem, rather than predefining a huge tree size limit deemed sufficient for generating a solution, a more focused decomposition can occur and a more efficient usage of computation could be expected. Bloat could also be reduced as a confounding factor.

Comparing the average tree size of standard genetic programming and layered learning reveals that they have the same average of 29 nodes. This indicates that using layers does not reduce tree bloat. Having a layer node encapsulating the kicking behavior should allow for a reduction of the minimum size of a solution tree. The tendency for program trees to increase in size as evolution continues is not affected by the presence of an encapsulated layer node. A correlation of tree size to goals scored for standard genetic programming gives -0.16, and for layered learning gives -0.07. In effect there is no relationship between the skills of the player and the size of its program tree.

#### E. Behavioral Taxonomy

One key element of soccer playing is finding the ball. The better the player is at finding the ball, the more opportunities there are for dribbling to the goals and scoring. As a result there is a clear evolutionary advantage to maximising the amount of the field that the player sees. The most complex behavior was found in experiments using the layer encapsulation. The player would move from its initialisation position towards the center of the opposition goal, zigzagging as it went (Figure 7). The closer it got to the goal, the tighter the zigzagging until it would seem to be spinning as it edged closer to the goal. When it reached the goal, it would turn around and run to its own goal. If at any stage the player

would see the ball, the player would move to it and attempt to score a goal. Due to the extra requirement of evolving the goal offset, this ball finding or field coverage adaptation was not the sole determinant in playing ability. Regrettably, this highly complex field searching behavior would often time out in the training and test runs before it would find the ball or before it could dribble the ball to the goals for a goal kick.

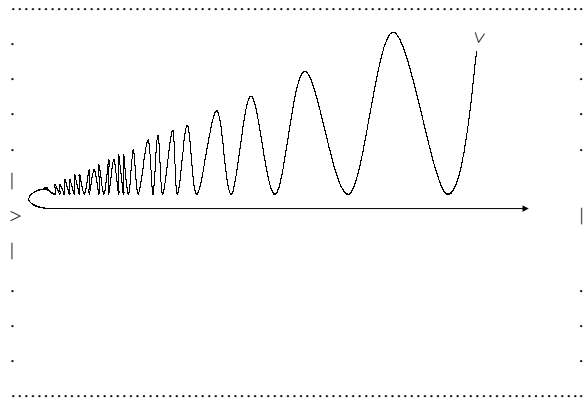


Fig. 7. Search path for the zigzagging player.

The fact that this complex behavior was only found in the encapsulated layered runs is evidence that more complex behavior is being evolved given the same computation parameters. This is made possible by the reduced effort of not having to evolve the goal offset again and that there is more space in the tree for behavior of greater complexity to develop, as the kicking behavior is encapsulated and only appears as a single node.

## VII. CONCLUSION

In this research we determined that layered learning is on average able to find evolve goal scoring behavior comparable to standard genetic programs more reliably and in a shorter time. Despite these advantages, the quality of solutions found by layered learning did not exceed those of standard genetic programming. Layered learning in this fashion requires a large amount of domain specific knowledge and programmer effort to engineer an appropriate layer and the effort required is not justified for a problem of this scale.

Layered learning did not exceed the quality of solutions produced by standard genetic programming but instead produced solutions of equivalent quality. Differences in overall goal scoring ability were shown to be statistically insignificant. The layered runs were shown to produce players of greater kicking accuracy.

Encapsulation with maintaining the population between layers was shown to be the best method for integrating the layer into continuing evolution. This was the most computationally efficient and led to solutions of high quality with greater reliability than any other layered method. It was demonstrated,

however, that this method is not guaranteed to integrate the layer in to the solution and this can lead to suboptimal solutions in a small portion of the runs.

Several improvements in computational efficiency were identified. The learning that takes place with layered evolution scores more goals in training than standard genetic programming. Players are evolved more reliably, with a greater percentage of layered learning runs scoring goals compared with standard genetic programming. Given the same amount of computation and maximum tree depth, layered learning allows for behavior of greater complexity to evolve. An improvement in efficiency was also evident with the average number of generations to find the best solution in a layered learning run being on average less than that of the standard genetic programming.

While layered learning with encapsulation provides the opportunity for an architecture of an equivalent size to learn a more complex behavior, thereby increasing the potential for scalability due to the encapsulation of complex behavior in a single node, this was not generally supported in the results. Parsimony of tree structure was not shown to have improved, as there was no correlation between the size of the program trees and solution quality, evidence that bloat remains a problem for scalability and efficiency.

Considerable degradation in evolutionary learning and efficiency is evident in the case that the layer utilises a vector in calculations which is incompatible with further learning.

## VIII. FURTHER WORK

An extension of this research into multi-player goal scoring scenarios with many more layers could prove worthwhile. An application to other appropriate problems would also be enlightening.

A more complex evaluation of the learning that occurs in each layer might produce consistently better results. If a pool of layer solutions are kept and a more exhaustive tournament held between them for inclusion in the next layer, this might encourage more generalisable and fitter behavior.

An investigation into the reasons for the apparent learning differential in goal scoring behavior between layered learning and standard genetic programming, which is not subsequently expressed in the quality of the best found in the runs, warrants further investigation.

As the learning is sensitive to the placement of the ball and player on the field, some training scenarios might be more conducive to evolving generalised behavior than others. This could be examined by competitively evolving field initialisations within a layer concurrently with the player behavior.

The number of generations before switching is set at seven hundred for these layered runs. A better method would be one that switched layers once an individual had achieved a particular threshold.

The scenario used here might be solved more efficiently using *concurrent layered learning* [14], allowing any defects in the learnt layer to be improved during further learning.

## ACKNOWLEDGMENTS

Software developed under VPAC (Victorian Partnership for Advanced Computing grant number EPPNRM054) was used in this work. The runs were performed on the VPAC Linux cluster.

## REFERENCES

- [1] Robocup official site. <http://robocup.org/02/02.html>, 2003.
- [2] M. Matkovic. Co-evolving competitive behaviors in genetic programming, October 2002. Honours thesis, Dept. of Computer Science, Royal Melbourne Institute of Technology.
- [3] S. Luke. Genetic programming produced competitive soccer softbot teams for robocup97. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference (GP98)*, pages 214–222, University of Wisconsin, Madison, Wisconsin, USA, 1998. Morgan Kaufmann.
- [4] Steven M. Gustafson and William H. Hsu. Layered learning in genetic programming for a cooperative robot soccer problem. *Lecture Notes in Computer Science*, 2038:291–301, 2001.
- [5] Shimon Whiteson, Nate Kohl, Risto Miikkulainen, and Peter Stone. Evolving keepaway soccer players through task decomposition. In E. Cantú-Paz, J. A. Foster, K. Deb, D. Davis, R. Roy, U.-M. O’Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M. A. Potter, A. C. Schultz, K. Dowsland, N. Jonoska, and J. Miller, editors, *Genetic and Evolutionary Computation – GECCO-2003*, volume 2723 of *LNCS*, pages 356–368, Chicago, 12-16 July 2003. Springer-Verlag.
- [6] V. Ciesielski, D. Mawhinney, and P. Wilson. Genetic programming for robot soccer. In A. Birk, S. Coradeschi, and S. Tadokoro, editors, *RoboCup 2001: Robot Soccer World Cup V*, volume 2337, pages 319–324, Seattle, WA, USA, 2002. Springer.
- [7] P. Stone. *Layered Learning in Multiagent Systems*. MIT Press, 2000. A Winning Approach to Robotic Soccer.
- [8] William H. Hsu and Steven M. Gustafson. Genetic programming and multi-agent layered learning by reinforcements. In W. B. Langdon, E. Cantú-Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 764–771, New York, 9-13 July 2002. Morgan Kaufmann Publishers.
- [9] Jay F. Winkler and B. S. Manjunath. Incremental evolution in genetic programming. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Genetic Programming 1998: Proceedings of the Third Annual Conference*, pages 403–411, University of Wisconsin, Madison, Wisconsin, USA, 22-25 July 1998. Morgan Kaufmann.
- [10] Alan D. Blair and Elizabeth Sklar. Exploring evolutionary learning in a simulated hockey environment. In Peter J. Angeline, Zbyszek Michalewicz, Marc Schoenauer, Xin Yao, and Ali Zalzala, editors, *Proceedings of the Congress on Evolutionary Computation*, volume 1, pages 197–203, Mayflower Hotel, Washington D.C., USA, 6-9 1999. IEEE Press.
- [11] Jeff Riley. Simple soccer simulator. RMIT University, Melbourne Australia.
- [12] Tucker Balch. ASCII soccer simulator. <http://www-2.cs.cmu.edu/trb/soccer/>.
- [13] J. R. Koza. *Detailed Description of Genetic Programming*, chapter 6. MIT Press, 1992. Genetic Programming: On the Programming of Computers by Means of Natural Selection.
- [14] S. Whiteson and P. Stone. Concurrent layered learning, 2003.