CHAPTER VII

EMERGENT GOAL-DIRECTED BEHAVIOR

7.0 Emergent Goal-Directed Behavior

The task environment is an important component of emergent intelligent systems. Often in computational problem solving, the feedback to a method comes from an objective function. Typically, these objective functions include an omnipotent teacher or exemplar for the task to compare the progress of the problem solver. Objective functions are also routinely used in the fitness functions of evolutionary algorithms. While global objectivity is easily provided in a fitness function when evolving solutions for simple numerical optimization problems, it is often impractical for problems with higher complexity. The difficulty stems directly from the objectivity of the fitness function, since objective accuracy often comes only at the cost of significant knowledge about the search space.

This chapter demonstrates that a population of learners in a competitive task environment provides the same or better benefits as a task environment that uses an objective measure. This is demonstrated empirically using the GLiB program to discover lisp programs that play Tic Tac Toe. However, instead of including an expert strategy in the fitness function to represent the task environment, as was done in the last chapter, the task environment uses only a metric that determines which of two population members is the better player. As a result, the population moves towards more complex strategies for the task. This apparent goal-directed behavior emerges from the direct interaction of population members in the task environment. Generalization in the resulting programs occurs due to the diversity of strategies in the population. The chapter begins with an in-depth look at the benefits and problems of competitive learning, describes why competition often biases learners toward sub-optimal solutions, termed *cooperative stable states*, and demonstrates how a tournament run over the population removes this difficulty.

7.1 Competitive Learning

Competitive learning is a long standing topic in machine learning (Samuel 1959; Tesauro 1992). A competitive learning process encourages an incremental development so that as new strategies are developed by one learner, its opponent adjusts its abilities and discovers new strategies of its own. This strategic "arms race" ideally increases the overall ability of the learners until they reach near optimal abilities. Interest for using competition in machine learning tasks stems from a desire for a program to discover the strategic nuances of a complex task directly from the first principles of interaction. Appropriate complex structures arising purely from the "physics" of the task environment would be the ultimate validation of machine learning capability. Such is the essence of emergent computation (Forrest 1991).

A natural collection of tasks on which to study competitive learning is the induction of strategies to play games, since these tasks typically require at least two competitors. Samuel (1956; 1966) was the first to use a self-practicing model in his checkers learning experiments. His program induced a linear function that predicted the utility of particular checker board configurations. He reports that a naive version of competitive learning in this system was unsuccessful. Samuel (1966) created a more complicated competitive method with two competitors that updated their respective evaluation functions at differing rates. Even though this more complicated method uses two separate learners, it is still considered competitive since the two learners comprise a single composite learner that is teaching itself.

Backgammon is a another game investigated with competitive learning. Tesauro (1992) describes a connectionist network, called TD-Gammon, that acquires an evaluation function that plays a very good game of backgammon. TD-Gammon uses it's developing network as both opponents to play a game. Once the game is completed the weights of the network are then updated using a temporal difference method (Sutton 1988). The game provides both positive and negative feedback to the network since it plays both sides. Tesauro reports that TD-Gammon plays better than his previous best network backgammon player, Neurogammon (Tesauro 1990).

The crux of competitive learning is the assumption that a learner improves itself through practice with itself, identifying flaws in its inductions that even it can exploit in the task. Once repairs are made, the learner moves closer to an adequate concept without a

priori knowledge signaling in which direction to move. Repeating competitive learning, it is thought, incrementally improves performance and progresses the learner through improving strategic phases. In essence, competition is expected to ratchet the performance of the learner toward more and more sound strategies. As each new strategic induction is made, it is employed both to its own benefit and demise, fueling further refinements to the concept.

While the thesis of competitive learning appears sound, the assumption that the learner steadily improves toward the goal is overly simplistic. Epstein (1992) argues that a self-competing learner alone does not provide sufficient feedback to acquire adequate concepts in some learning environments. Her experiments on two player deterministic games, such as Tic Tac Toe, shows competition acquires concepts consistent in ability as those induced using a teacher playing optimally about half of the time and randomly otherwise. She concludes that competition is a bit myopic about the identification of incorrect inductions in the learner. This creates the potential to underexplore the space unless there are specific attributes in the task to encourage otherwise.

In any learning system, the implicit "goal" is to locate a strategy such that no feedback received forces the training algorithm to change it. In other words, *the concept is stable with respect to the teacher and method of credit assignment*. Thus, the ratcheting of performance to a good strategy is only one possible limit situation for a competitive learner. Another training sequence could lead to limit performance that is optimal for the player playing itself, but fails drastically when forced to play other players. For instance, in a deterministic game such as Tic Tac Toe, a learner could draw itself consistently without being even remotely proficient at the game, in effect "cooperating" with itself to maximize its score and minimize its own modification. Such a situation, where the learner identifies a sub-optimal strategy that is stable with respect to the task environment and the method of credit assignment is termed a *cooperative stable state*.

In order for competition to be an effective general method, it's bias toward cooperative stable states must be curbed. But to curb it we must understand exactly what is causes these sub-optimal states. Axelrod has studied cooperation in a diverse collection of competitive environments (Axelrod 1984). He argues that for cooperation to arise, an environment must provide a high probability for repeated meetings and allow both players to reciprocate for past defeats. In the standard, naive definition of competitive learning, the

learner is always its own opponent. This creates an environment where the learner is assured of seeing the same opponent, itself, again each competition. Self competition unwittingly creates a preference for any concept in the space that minimizes the changes in itself after competition on the task.

Thus the reflexive environment of Tesauro's (1992) TD-Gammon should *maximize* the potential for the network to fall into poor strategic minima. Tesauro (1992) and Epstein (1994) argue that the success of TD-Gammon is due in part to the non-determinism inherent in the task. Because backgammon requires a dice roll on every move, the learner is forced into a wider variety of strategic situations than if the task was deterministic. While this conclusion is basically correct, it is incomplete. The dice in backgammon discourages TD-Gammon from settling into cooperative stable states by creating unpredictable advantages for one player, reducing the competitor's ability to reciprocate. The roll of the dice occasionally forces play into board configurations that have never been visited in any previous game and consequently provides feedback to refine the network. Even if TD-Gammon discovers a suboptimal cooperative solution, the non-determinism in the task elicits feedback that moves the network away from a cooperative stable state by forcing a change in its structure.

However, the non-determinism of the dice is a weak remedy especially in light of the network being assured of playing itself again. The attraction of cooperative stable states in some tasks is strong enough to counter the learning gradient supplied by non-determinism. One reason Tesauro's network may have taken so long to train is the gradient toward cooperative stable states and the feedback provided by competition on the non-deterministic task worked to cancel each other's effects.

7.2 Competitive Learning in Evolutionary Algorithms

In evolutionary algorithms, the population represents a natural source of diversity that, while not entirely random, can be recruited to create virtually non-deterministic competitive environments. Competitive populations have been under-exploited in evolutionary algorithms; exactly why is unclear. One possibility is that competitive environments are thought to be too unstructured to guide a population toward a particular goal unless the goal is suitably vague or non-existent. Such an attitude could explain the relegation of competitive evolutionary environments to the Artificial Life community (e.g., Ray 1992;

Lindgren 1992). Another possibility is that competition is considered too expensive for practical problems; that it requires too many evaluations of population members to determine an accurate ranking. Without an accurate enough ranking, the natural dynamics of the evolutionary process might be compromised.

7.2.1 Standard Fitness Functions and Competitive Selection

The standard fitness functions used in genetic algorithms are exemplified by the functions studied in De Jong (1975). Such functions return the same fitness for an individual regardless of what other members are present in the population. Their independence from the population's composition allows these functions to provide an accurate and consistent fitness measure throughout the evolutionary process.

While global accuracy is easily computed when evolving solutions for many simple optimization problems, it is often impractical for problems with higher complexity. The difficulty stems directly from the objectivity of the fitness function, since objective accuracy often comes only at the cost of significant knowledge about the search space. For instance, consider the expense of a standard fitness function for evolving an optimal strategy for a particular game. Such a function would need to test members of the population against all possible strategic situations to garner an objectively accurate measure. For anything but a trivial game such a computation is immense. If a suitable "expert" strategy were available, an independent fitness function could still be constructed, however, the evolved solutions would only be "optimal" with respect to this "expert" rather than the original task.

Tournament selection, a competitive selection method, is currently popular in genetic algorithms (Goldberg 1989a) and evolutionary programming (Fogel 1992a). Once the fitness of each individual is discovered, the parents of the next selection are selected through comparison. In a *k-competition* in genetic algorithms (Goldberg 1989a), the fitness of k randomly chosen members of the population are compared. The competitor with the highest fitness value is declared the winner and copied into the next population. This process is repeated with replacement until the next population is filled. In the tournament selection of evolutionary programming (Fogel 1992a), a competition score for each population member is created by comparing its fitness with *k-1* other population members and awarding a point for each competitor with a lower fitness. After each population member is scored,

those members with the highest competition scores become the parents of the next population.

Note that both competitive selection methods still require an objective fitness function and thus are still susceptible to the associated difficulties described above. In order to remove the reliance on objective fitness functions, the competition must be used to determine the fitness of the individuals in the population.

7.2.2 Competitive Fitness Functions

In contrast to standard fitness functions, a *competitive fitness function* is any method for calculating fitness that is *dependent* on the current population to any degree. The dependency can be relatively minimal, such as on a single population member, or fairly comprehensive in functions that use the entire population to determine a single strategy's fitness. In essence, competitive fitness is the original intention behind a fitness function since it provides a measure of an individual's ability relative to the current population rather than relative to the global optimum.

This following sections describe two previous models of competitive fitness: a full population competition described in Axelrod (1987) and a bipartite competitive model described in Hillis (1992). How these respective methods address cooperative stable states is also described.

7.2.2.1 Axelrod's Full Competitive Model

Axelrod (1987) experiments with both an independent and a competitive fitness function to evolve strategies for the Iterated Prisoner's Dilemma (Luce and Raiffa 1957; Rapaport 1966; Axelrod 1984). For the independent fitness function, Axelrod uses a weighted sum of the scores against eight pre-selected strategies, where the weights and representative strategies were selected on the basis of knowledge gained from previous experiments (Axelrod 1984). In the competitive fitness function, Axelrod (1987) tests every population member against every other population member, which presents an adaptive developmental environment for the population. However, there is no mention in the study about the relative abilities of the evolved strategies. A schematic of the competitive pairings associated with this method appears in Figure 34a. Assuming the size of the population is n, the number of competitions executed in a generation is n^2. When the task to be solved is quite

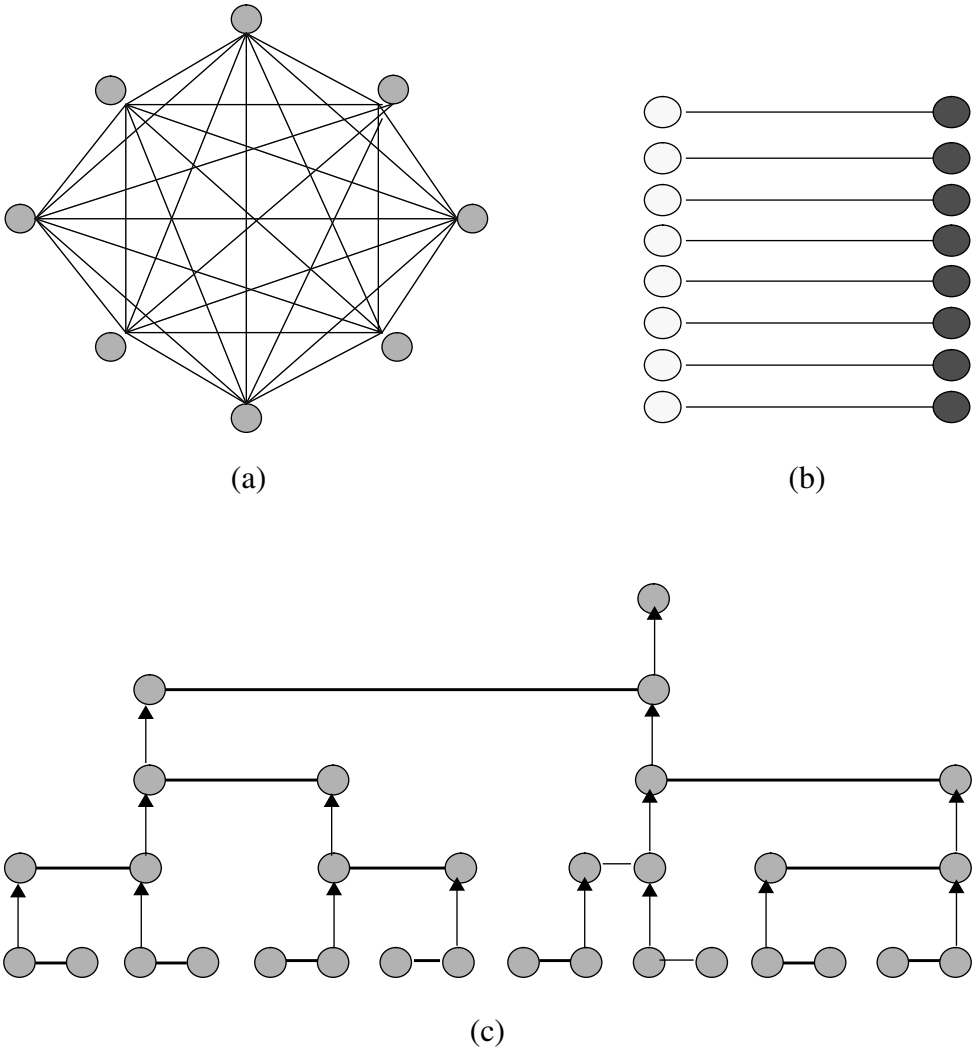

Figure 34: *Three types of competitive fitness functions. (a) Full competition used in Axelrod (1989); (b) Bipartite competition used in Hillis (1992); and (c) Tournament fitness with each horizontal line designating a competition and each upward arrow designating the winner progressing in the tournament.*

complex and requires a large population or a significant number of generations, this number of competitions per generation may be prohibitive.

Unfortunately, Axelrod's experimental goal was to encourage cooperation on the task rather than discourage cooperative stable states. Axelrod's experiments attempted to determine if a cooperative strategy for the Iterated Prisoner's Dilemma could evolve from random initial conditions in the population. Because each member of the population always competes against every other member of the population, there is ample opportunity for reciprocation between differing strategies. In fact, it was exactly this reciprocation that Axelrod was interested in inducing. Thus, when the object is to avoid cooperative stable states, a full competition may affect the search adversely.

7.2.2.2 Hillis' Bipartite Competitive Model

Hillis (1992) demonstrates a competitive fitness function with an interesting approach. The problem explored in Hillis (1992) is to evolve a sorting network for any arrangement of 16 integers with as few element exchanges as possible. Notice that this task is not so different from a game; the sorting networks represent various strategies and the 16! potential arrangements of integers represent the various board configurations. Clearly, using a fitness function that tests all possible permutations on each sorting network is impractical. Additionally, a static subset of permutations would clearly encourage solutions that sort only the chosen subset. Hillis (1992) reports that even using a randomly selected subset of permutations that changes every generation does not provide a sufficient environment to evolve adequate sorting networks.

In order to maintain a consistently difficult set of permutations to evaluate the sorting networks, Hillis (1992) creates a second population. Each member of the second population encodes a small set of permutations of integers to be sorted by one of the sorting networks of the first population. In this experiment *both* populations evolve from generation to generation. Fitness for the population of sorting networks is defined to be how well the member sorts the various permutations within the associated member of the second population. The fitness of a member in the second population is a measure of how *poorly* the sorting network sorts the set of permutations it contains. This bipartite competition is illustrated in Figure 34b. With this fitness function, Hillis' system evolved a sorting network

with only 61 position exchanges, which is a single exchange worse than the best known sorting network for 16 numbers.

Assuming the sizes of the populations are the same and when combined equal n, the bipartite competition in Hillis (1992) uses a total of $n/2$ competitions each generation. This is far fewer than a full competition, as in Axelrod (1987). However, while the fitness function used in Hillis (1992) is an example of a competitive fitness function, there is no method for determining which member of the population is the best sorter. Because each sorting network competes against a single member of the second population there is no basis of comparison *between* sorting networks. The score received by a sorting network is relative to the difficulty of permutations it attempted and each sorting network sees distinct sets of permutations. In addition, the bipartite nature of the competition model used in Hillis (1992) may be unnatural for some problems.

Pitting evolving members of a population against each other to determine fitness creates an interesting tension in evolutionary algorithms. For instance, while the population of sorting networks in Hillis (1992) is adapting to the specific permutations it is being tested against, the population of permutations is searching for the set that forces the sorting networks to perform as badly as possible. In order for the sorting networks to reproduce from generation to generation consistently, they must generalize their sorting ability rather than encode for a specific subset of permutations. The need to compensate for the continuing diversity in the permutations directs generalization in the sorting networks.

Hillis' bipartite model for competition does not remove the bias toward cooperative stable states. Consider a population as a single, very complex learner comprised of a number of homogeneous components, and, as in Samuel's (1959) competitive learner, the two populations as a single learner. At this level, Hillis' system is just an implementation of standard competition with a more complicated learner. Therefore, similar to above, Hillis' learner seeks out cooperative stable *populations*, i.e. populations that, on average, don't force each other to change significantly. The entire population need not be involved to slow learning, only enough to make the current population relatively stable with respect to itself. Thus, a population of learners alone is insufficient to inhibit cooperation. Note that this type of cooperation was the object of Axelrod's (1987) experiments.

7.3 Tournament Fitness

This section describes *tournament fitness*, a third model of competition in a fitness function that addresses the problem of cooperative stable states. Tournament fitness uses a single elimination, binary tournament to determine a relative fitness ranking for the population. Initially, the entire population is in the tournament. Two members are selected at random to compete against each other with only the winner of the competition progressing to the next level of the tournament. Once all the first level competitions are completed, the winners are randomly paired to determine the next level winners. The tournament continues until a single winner remains The fitness of a member of the population is then its height in the playoff tree, the player at the top being the best player of the generation. The competitive pairings for tournament fitness are illustrated in Figure 34c. The hierarchical nature of the ranking is strictly enforced, ties being broken by random selection. In the case that the number of competitors at a level is odd, a single population member is passed to the next level of the tournament without a competition, effectively receiving a "bye" for that round. The total number of competitions for a population of size n is:

$$\sum_{i=1}^{\lceil \log(n) \rceil} \left\lceil \frac{n}{2^i} \right\rceil = n - 1 \qquad \text{(EQ 25)}$$

which is one fewer comparison than required to play each member of the population against a single "expert" strategy in a comparable independent fitness function.

Quantification of performance on the task is unimportant when using tournament fitness; all that is required is a concept of "better" to compare two strategies. This removes all need for determining exactly how much better one player is than another - the resulting tournament hierarchy is sufficient information for reproduction. Unless the competition, i.e., the measure of "better", is noisy, an optimal player always wins the tournament. However, if the environment is suitably complex and an optimal strategy is not in the population, it is possible for an average or even a comparatively poor strategy to win the tournament for a particular generation. Thus this competitive fitness function can contain a level of noise associated with its ability to rank any given population. How accurately the tournament ranks the population is dependent upon the set of competitors met. For instance, if the best player in the population competes in the initial round of the tourna-

ment with the second best member, only the best player moves up the hierarchy with the second best player being assigned the minimal fitness.

Fortunately, the inherent noise of tournament fitness functions is not a serious problem given that the fitness ranking is being created to decide the proportions for reproduction. Consider that the worth of a single competition, in terms of reproduction, is inversely proportional to how high in the tournament the competition occurs. In other words, the higher the level of the competition, the less it is worth in terms of reproductive advantage. For example, consider a single competition in the initial round of the tournament. The first competition determines which half of the rankings the two competitors reside. The loser receives a fitness that places it in the lowest 50% of the population's ranking. The winner earns a fitness value at least in the upper 50% which gives it a reasonable chance for reproducing. With each successive round of competitions, exponentially decreasing subsets of the population are divided into winners and losers until the last competition decides between best and second best for the generation. At this level, the difference in the probability of reproducing is negligeable with any reasonably sized population.

Once a tournament has been run, any standard selection method can be used to designate parents for the next generation. Because all the population members that lost at the same level of the tournament have the same fitness values, tournament fitness naturally deemphasizes their worth relative to each other. This is more beneficial than over-committing to an erroneous complete ordering of the population. Selecting between members with the same fitness must be at random which promotes better mixing of the representational components when using crossover and discourages premature convergence. In fact, in many situations tournament fitness naturally discourages convergence since as a particular strategy becomes too numerous it is forced to literally compete against copies of itself. This is akin to a predator/prey system where the prey has been hunted to such low population levels that the predators are forced to feed on each other.

Tournament fitness sets up the same oppositional tension as in Hillis (1992), but in a more comprehensive manner and requires only a single population. Because it is unlikely that all strategies have been represented in past populations, complex developing strategies may contain flaws that may be exploited by a variety of simpler players. This is much like a relatively good chess player being beaten by a novice who only knows the "fool's mate" strategy. Since for any complex strategy there may be numerous ways for it to be

beaten, a complex ecology of strategies for the task can develop during the course of the run with only the most robust strategies consistently appearing at the top of the tournament. These complex strategic ecologies are similar to those described in Ray (1992) and Lindgren (1992) but are relative to the specific task being solved.

Tournament fitness also reduces the risk of falling into cooperative stable states. Because any one learner is playing at most $\log(n)$ other players, where n is the population size, the chance of two learners meeting in consecutive tournaments is low. By the time learners are rematched, they are modified enough by the learning mechanism to avoid cooperation. If the same learners do meet consistently at the same level, then it must be at a level where the chance of matching the pair is high which can only happen close to the end of the tournament for the generation. If two learners consistently meet at a high level of the tournament, they must be consistently beating most of the other strategies in the population, making them necessarily robust on the task, assuming the population is fairly diverse.

Another beneficial feature of this competitive model is that the "worth" of a single competition is dependent on the level of the hierarchy at which the opponents compete. A pairing that occurs at the first level is extremely significant since it places the loser in the bottom half of the of the ranking and the winner in the top half. The worth of games higher up in the tournament decrease because they divide between exponentially decreasing subsets of competitors. In other words, it makes a bigger difference reproductively between being in the top or bottom half of the population than it does between being in the top 1 percent or top 2 percent. Axelrod (1984) suggests one method for breaking reciprocity is to change the game's payoff function so that if competitors do compete and attempt to reciprocate, their respective scores can not balance out. Tournament fitness' hierarchical ranking of the population does exactly this.

7.4 Experiments

To test the tournament fitness function, several experiments were run using the Genetic Library Builder (GLiB) (Angeline and Pollack 1993a; Angeline and Pollack 1992), described in chapter 6 but modified to perform a hierarchical tournament as described in the previous section. The task in these experiments was again Tic Tac Toe and the representation language for the evolving programs was also the same. The only differ-

ence between the experiments of this chapter and those of chapter 6 involve the type of task representation used in the fitness function.

To compare the ability of the programs evolved by standard independent fitness functions and tournament fitness, a collection of "expert" TTT players of varying strategic ability were used. The three experts used in this experiment were RAND, NEAR and BEST. RAND simply chooses a legal position at random to determine its move for a given board configuration. At the other extreme is BEST which chooses the optimal position to play on each move. No strategy, evolved or otherwise, can win a game against BEST. NEAR, the third algorithm, is the same expert strategy used in the experiments of chapter 6 except the adaptive element was removed. NEAR still performs near optimally except that it can be forked by its opponent. Unless forked, NEAR either draws or beats its opponent. Both NEAR and BEST non-deterministically choose between equal moves in order to force more robust play from the developing programs.

The following experiments cover four different learning situations. The first three use an independent fitness function consisting of one of the above experts to evolve modular programs for the Tic Tac Toe task. A single competition against an expert is scored by the number of moves the evolved program makes with a 5 point bonus for a draw and a 20 point bonus for a win. A program's average score over four games is its fitness in these experiments. These runs represent a range of objective fitness functions that might be used for this task. The final experiment uses the tournament fitness function described in the previous section and is labeled POP in the results. Scoring a single competition between two programs is as described above. A program is deemed "better" than its opponent if it had the greater score after two games, with each player taking the first move in one game. As described in the last section, if the scores are equal the winner of the competition is chosen at random.

Each of the experiments use a population size of 256 and run for a total of 150 generations. All experiments use roulette wheel selection (Holland 1975) with linear scaling (Goldberg 1989a) and a scaled fitness maximum of two. Other than the fitness function, all other factors and parameters are identical for the four runs.

7.5 Results

In order to compare the training effectiveness of each of the four fitness functions, the evolved program with the best fitness from each experiment plays 2000 games against each of the three designed experts. The distribution of wins, losses and draws provide a crude mesaure for comparison between the acquired abilities of the evolved programs. These results for the four experiments are shown in Table 7. As can be seen from the table,

Table 7: *Table showing results of competition experiments.*

Fitness Function Used	*Evolved Program vs. RAND*			*Evolved Program vs. NEAR*			*Evolved Program vs. BEST*	
	Wins	Draws	Losses	Wins	Draws	Losses	Draws	Losses
RAND	1125	0	875	0	0	2000	0	2000
NEAR	802	104	1094	144	123	1733	0	2000
BEST	310	535	1155	0	360	1640	0	2000
POP	781	471	748	61	588	1351	481	1519

none of the evolved modular programs induced the optimal TTT program. This is due in part to the low level primitive language and the insistence that GLiB acquire both control and evaluation information in the same program. More interesting is the difference in ability of the programs induced using the three expert algorithms. The program evolved using RAND is fairly poor, only able to beat the RAND expert a little more than half the time while appearing totally incompetent against the better experts. NEAR's evolved protege embodies a more able concept that displays a broader ability to compete. The program acquired using BEST appears to have induced the ability to draw opponents in many situations but is weak in its ability to win, even against RAND.

Of even more interest is the final experiment which uses tournament fitness. The program induced in this experiment is clearly a more robust player if not more proficient than

any of those induced using one of the experts. The fact that the program evolved by tournament fitness was the only evolved program that could draw BEST, and did so fairly frequently, shows it acquired a more sophisticated algorithm for TTT than those induced by the independent fitness functions.

An important question raised by these experiments is why none of the programs evolved using the independent fitness functions could draw BEST. This is straightforward if the quality of environment presented by each fitness function is considered in turn. First, RAND provides no pressure for a program to induce a complex winning strategy. Simply hard coding only three positions in a row guarantees the program will win a few games against RAND. But such a strategy is easily thwarted by NEAR and BEST. When using BEST as the strategy in the independent fitness function, no program ever receives positive feedback for making three in a row, and consequently none of the evolved programs induce this ability very broadly. Programs perform well against BEST when they can play almost anywhere on the board and maximize the number of moves they make before being beaten. This translates into a program that plays several moves but can't put those moves together to form a winning combination. When NEAR is used to evolve solutions, the programs emphasize setting up forks, since this is NEAR's only flaw and the scoring function emphasizes wins strongly. Once a program evolved using NEAR is able to fork it and win, there is no reason to improve and little pressure to develop the ability to consistently draw an opponent. This type of strategy is sufficient to perform well against RAND, but is easily defeated by BEST which protects itself from being forked.

On the other hand, the program evolved by tournament fitness is forced to play against a number of differing strategies from the collection of strategies developing in the population. Some of these strategies are simplistic and provide little difficulty while others are equally competent on the task. Still other strategies in the population are fairly simple but exploit specific weaknesses in the population. One measure of the complexity and diversity of the population is how often simpler strategies, the strategic underdogs, defeat more complex strategies. However, it is difficult to observe this phenomena since there is no objective measure for the strategic ability of a TTT strategy.

In order to approximate an objective measure of the complexity of TTT strategy, this experiment defines a measure from the variety of pseudo-objective fitness functions. Simply summing the values returned from the RAND, NEAR and BEST fitness functions sup-

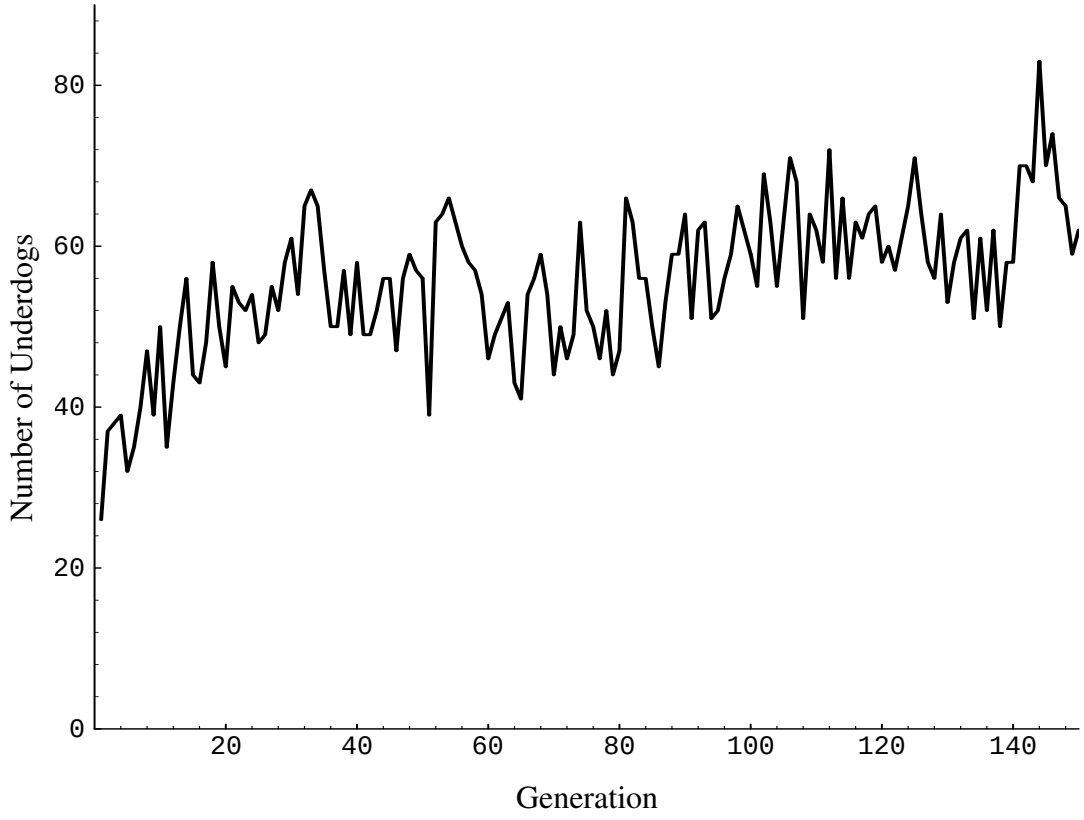

Figure 35: *Number of underdogs that won their competition for each generation of the run.*

plies an adequate approximation for the relative abilities of the evolving programs. Using this measure of complexity, Figure 35 graphs the number of times an underdog strategy, i.e. one determined to be less complex, defeated a more complex strategy for each generation of the run using tournament fitness. The graph shows that the number of underdogs that win their competition increases over the course of evolution peaking at about 33% of the competitions in the tournament towards the end of the run. This indicates that the ability of the population to create strategies to exploit specific strategic flaws increased over the run. Figure 36 shows the difference in the complexity measure values for the underdogs that won their competitions for each generation. The solid line shows the average difference in objective scores for all underdogs that won in that generation. The dashed line graphs the maximum difference in scores for underdogs that won over the course of the

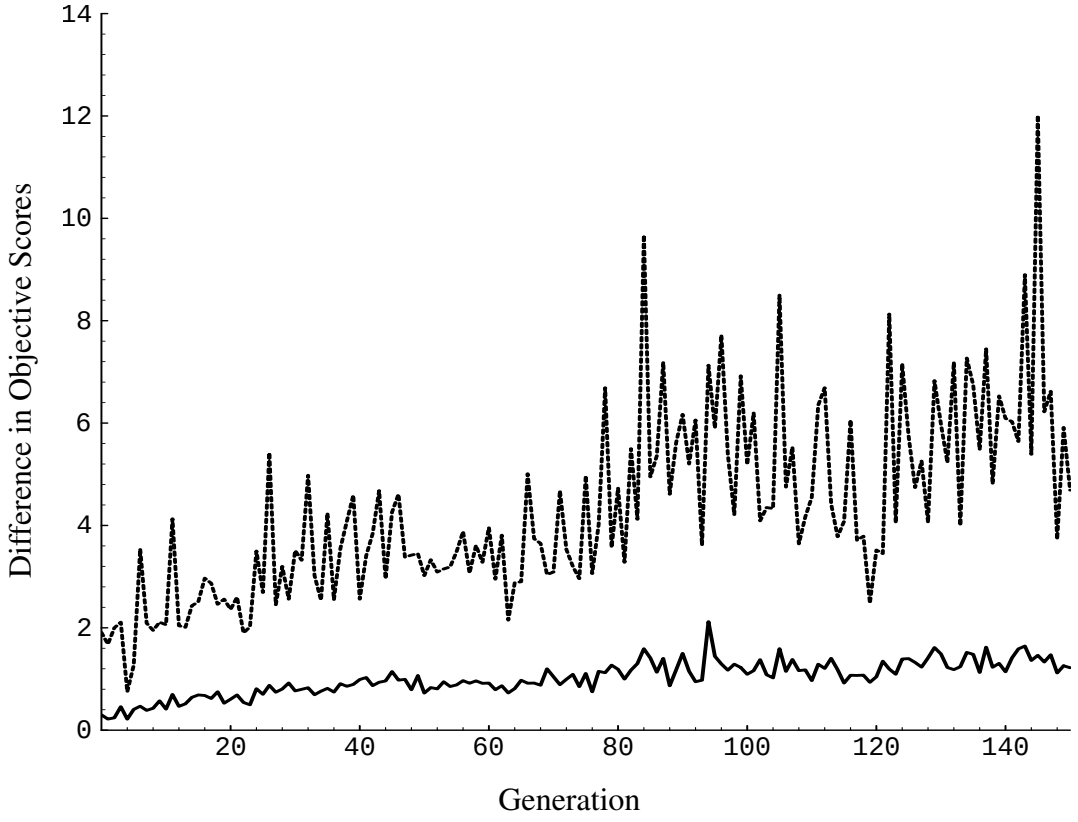

Figure 36: *Average (solid) and maximum (dashed) "objective" fitness difference between winning underdogs and their competitor.*

run. Both of these graphs show a marked increase suggesting that not only are more underdogs winning but the complexity of the strategies they are beating is increasing.

7.6 Discussion

One of the primary advantages of competitive fitness functions is their ability to adapt to the level of complexity of the population. In the experiment above, NEAR and BEST present difficult adversaries from the beginning and do nothing to identify preferred performance in the early populations. Such a large difference between the ability of the initial population and the strategy in the independent fitness function can inhibit the evolution of solutions if not compensated for in the function explicitly. For instance, in previous experiments with NEAR, the fitness function awarded a bonus for evolved programs that successfully blocked a win (Angeline and Pollack 1993a). With a competitive fitness

function, this problem is removed since the population is its own measure. As the ability of the individual members of the population increases on the task, the difficulty of the fitness function evolves with them. Since the function is dependent on the population, it tracks through the population's non-linear development without the need for measuring the average member's ability explicitly. Additionally, the open-endedness of a competitive fitness function is dependent on the open-endedness of the representation for the population members.

As stated above, ecologies of strategies develop in competitive fitness functions that provide a more consistently difficult environment than independent fitness functions. As evolution continues, the ecological balance shifts in the population to take advantage of exploitable strategic niches. This is indicated by the graphs of Figure 35 and Figure 36.The question still remains as to what prevents the population from wandering aimlessly through the space of strategies rather than moving towards more complex solutions. Given that the population maintains some level of diversity, this is straightforward. Because there are many differing strategies that could be met in any generation, only solutions that perform well against a number of them will consistently be in the upper tiers of the tournament and be able to continually reproduce. The constantly changing competitive environment forces the developing programs to generalize their abilities, as in Hillis (1992).

One advantage for tournament fitness over the bipartite competition used in Hillis (1992) is that proper counterexamples of various strategic difficulty are evolved within a single population removing the need for distinct populations to be maintained and separate fitness functions to be constructed. The adaptability of the content of the single population may be more beneficial to the evolutionary development of solutions than a predetermined bipartite population. Of course, which form of competition is appropriate for a given task is somewhat problem dependent.

7.7 Conclusion

In this chapter, it was argued that competitive fitness functions have many advantages over the independent functions that are typically used in genetic algorithms. Not the least of these advantages is that a competitive fitness function requires only a minimal understanding of the search space for a complex task. This removes the need for task knowledge that may be extremely difficult to engineer out of the problem. Furthermore, by employing

all strategies represented in the population as potential counterexamples, the fitness function automatically adapts to the nuances of both the individual problem and the specific progression of populations in a particular run. Finally, the experiments above and those in Hillis (1992) demonstrate that using the population as the reservoir for comparison is preferable to using an exemplar for the task when an objective measure of fitness is unavailable. Using a competitive environment permits the evolutionary process to acquire a more general solution that approximates global optimality relative to the task rather than relative to the provided exemplar.