Heterogeneity and Specialization in Evolving Teams

Terence Soule
Department of Computer Science
St. Cloud State University
St. Cloud, MN 56301-4498
tsoule@eeyore.stcloudstate.edu
(320)654-5183

Abstract

Cooperative teams can be used to improve the performance of standard genetic programming. However, designing appropriate team member choice and cooperation mechanisms for producing successful teams is an important and sometimes difficult task. This task is complicated by the type of programs being generated. For example, programs producing boolean valued outputs can use simple voting as a cooperation mechanism whereas programs with real valued outputs cannot. This research examines several team choice and cooperation mechanisms for linear regression problem.

The results suggests that cooperation produces the most improvement when the mechanisms favors specialization. The best results are achieved when each team member can focus on a relatively small portion of the problem domain and when the member's influence outside of that domain is relatively small. Specialization is found to be a function of both the team choice and cooperation mechanism.

1 Introduction

Recently there has been growing interest in using genetic programming (GP) to evolve cooperative behavior between programs [HSSW95, LS96, HS97, Sou99]. Most of the research in evolving teams has focused on problems requiring team solutions, particularly predator-prey pursuit problems and simplified versions of team sports. This research was fairly successful at producing teams that could cooperate to solve problems a single individual could not solve.

The success of cooperative, team approaches can be expanded to problems that do not inherently require a team to solve. Recent results have shown that cooperating teams can produce improved performance on the even-parity problem, a problem which does not require a team solution [Sou99]. In those experiments cooperation was achieved through voting. It was clearly demonstrated that the team members were cooperating. The teams' performance greatly exceeded the performance of either individual team members or individuals evolved outside of a team. The teams had the additional, unexpected benefit of producing more compact code. A team consisting of five voting members produced better solutions with smaller total code than did evolved individuals. Thus, there appears to be a clear benefit to using teams even on problems that don't inherently require a team solution.

In this paper we examine two fundamental team parameters: the member choice mechanism and the cooperation mechanism for the linear regression problem. The linear regression problem is particularly interesting because it requires a novel approach to cooperation. Simple majority voting is not possible with real valued solutions. We are interested in determining which mechanisms favor heterogeneous teams and specialized team members and which of these features produces the best results.

2 Cooperating Teams

The idea behind cooperating teams is that the effort of several individuals (the team members) can be combined to produce better results than would be achieved with a single individual. Several factors may produce improved results. Team members may specialize so that each member has a simpler subproblem to solve. Teams may also evolve more effectively because changing a single member of a team is less likely to destroy the team's performance than changing an individual.

There are three primary issues in using teams. How will the teams be chosen, what cooperation mechanism will they use and how will credit be assigned among the team members?

2.1 Choosing Teams

In general teams can be homogeneous, identical team members, or heterogeneous, different team members. Homogeneous teams are generally easier to implement. Typically, evolution is applied normally. When an individual is chosen for evaluation N copies of the individual are made (where N is the team size) and the N copies are evaluated as a team.

Homogeneous teams limit the possible cooperation mechanisms. For example, cooperation through voting is clearly not possible with homogeneous teams. Heterogeneous teams are somewhat more difficult to implement. However, most researchers have found that heterogeneous teams produce better results.

Several implementations of heterogeneous teams have been used. One approach is to evolve individuals normally, then, during evaluation, N individuals are chosen from the population to form the team. Thus, every team is different and each individual will be a member of many different teams over the course of a evolutionary run.

This approach is actually expected to produce fairly homogeneous teams for two reasons. First, after many generations the population has started to converge. Thus, a team is likely to consist of N versions of the same individual. Furthermore, evolution should favor individuals that cooperate well with themselves increasing the pressure to evolve homogeneous teams.

Second, during each evaluation the programs will be teamed with different members. Thus, they cannot 'expect' any particular behavior from the other team members. This environment should favor generalists, programs that can cooperate well with any other program.

To reduce the amount of homogeneity N separate populations can be used. During evaluation one individual is chosen from each population. This is more likely to produce heterogeneous teams because it avoids the first problem described above. However, the second problem still applies. So, a certain amount of homogeneity is expected.

In addition, this approach seems likely to favor rapid convergence within the N populations, although each population may converge to a different solution. An individual can not have any "expectations" of how its team mates will perform because they are draw randomly from the other populations. Thus, it may be difficult for individuals to evolve cooperative behaviors until the populations begin to converge making expectations possible. This will produce increased pressure for rapid convergence.

A third approach is to have each individual consist of N separate programs that act as single team. This allows heterogeneous teams without the additional convergence pressure to produce members with "expected" behaviors.

The drawback with this approach is that it requires a modification of the crossover operator. Standard crossover can not be used when each individual consists of several seperate programs.

Haynes and Sen have studied several of the possible crossovers [HS97]. They tested crossing a single randomly chosen member from one team with a single randomly chosen member of another team, crossing all members of one team with the corresponding members of another team, and using a crossover "mask" to determine the crossover members. Preliminary results suggest that crossovers between corresponding team members (i.e. member 1 of one team only crosses with member 1 of other teams) produces the best results, possibly because it favors specialization among the team members.

The number of members per team (the team size) is also a consideration. Very little work has been done relating team size to performance as most researchers have used problems with fixed team sizes, e.g. evolving a vollyball team (6 members) [RD94].

2.2 Cooperation Methods

To date almost all research involving teams has used an implicit cooperation mechanism. The most common approach has all team members acting in a common environment, cooperation occurs (or does not occur) by default. (This is sometimes referred to as evolving coordination rather than cooperation, but the principle of autonomous team members working towards a common goal is the same.) Examples of this approach include predator-prey problems with multiple predators [HSSW95] and evolving a vollyball team [RD94].

An alternative approach is to use an explicit cooperation mechanism. Each team member generates a solution, or portion thereof, and an explicit mechanism is used to combine the member's solutions. Iba used a variation of this approach. The individuals were evolved normally, as individuals, not team members. After the final generation the best individuals were

Table 1: Summary of the linear regression problem.

Objective	Find a curve fitting function (sin).
Terminal set	Input value X and random constants
Function set	+, -, *, and / (protected division)
Fitness	Square root of the sum of the squares of the error at 40 test points
Population size	1000
Crossover probability	80 percent
Mutation probability	5 percent
Selection	stochastic remainder
Termination criteria	100 generations
Maximum size of trees	none
Initial population	grow, no size limits
Number of trials	fifty

combined into a single team that voted on the solution [Iba97].

In earlier research used an explicit voting mechanism to evolve teams to to solve the even-parity problem [Sou99]. Teams consisted of five individuals that were evolved as a unit. The results clearly showed that cooperative behaviors evolved and improved the overall results.

2.3 Credit Assignment

The final task in evolving teams is assigning credit to the individual team members. The most common approach is to have all team members receive equal credit. This is in keeping with evolutionary computation's generally minimalistic approach to credit assignment and is used here. (Most evolutionary approaches give a single reward to the entire individual without attempting to proportionally reward more significant subprograms. Similarly, with teams the entire team is rewarded without attempting to proportionally reward more significant team members.)

3 Experimental Method

3.1 The Test Problem

The linear regression problem is used as a test problem for these experiments. The goal of the linear regression problem is to evolve a function to fit a set of data points. For these experiments the data points were generated by a sin function. The sin function was chosen because there is no perfect solution with the given operators. Thus, the GP can continously attempt to refine its solution and the results are not influenced by a population finding a perfect solution before the final generation.

The fitness is the square root of the sum the squares of the errors on 40 test points between $-2PI$ and $2PI$. The goal is to minimize this value.

For the linear regression problem the operators are addition, subtraction, multiplication and protected division. The terminals are real constants and the input variable. The constants are real numbers initially generated in the range -10.0 to 10.0. The mutation operator changes the constants by a real value in the range -0.5 to 0.5.

For elitism fifty copies of the best individual are made in each generation. However, these copies are allowed to undergo crossover with the normal 0.8 probability. During selection if two individuals have the same size the smaller is chosen. This introduces a limited form of growth control.

In the intial population programs are allowed to grow randomly without size or depth limits. This produces a fairly wide distribution of program sizes. Selection is rank based. Each individual is assigned a value equal to (population size-(rank-1))*2/population size (giving a value between 2 and 2/population size). The whole number portion of this value is the number of copies made, the decimal portion is the probability of an additional copy. Other details of the problem are shown in Table 1.

3.2 Team Choice

Two mechanisms for team choice were tested. The first mechanism randomly chooses N-1 individuals from a single population to act as the team members for the individual being evaluated. (As noted previously this approach is expected to produce fairly homogeneous teams and these trials are labeled homogeneous teams in the remainder of the paper. The cooperation mechanism used was median voting, described below.) The second mechanism evolved teams as individuals. Each individual in the population consisted of N distinct programs that form a team. This approach is expected to favor heterogeneous teams.

3.3 Cooperation Mechanisms

Three cooperation mechanisms were tested. The first mechanism averaged the output of the team members

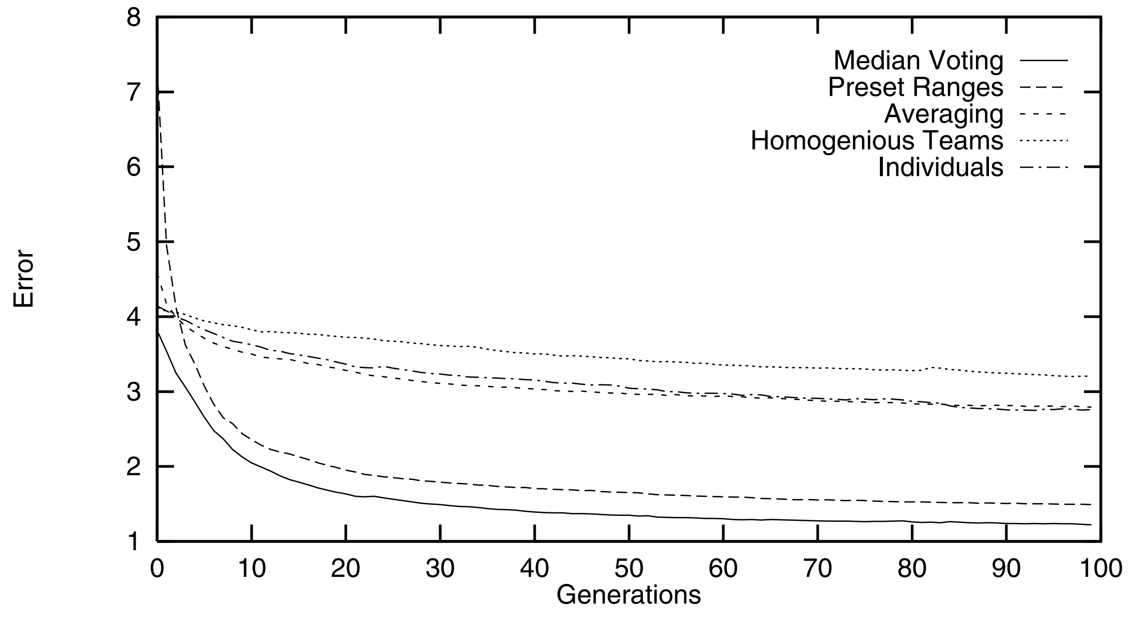

Figure 1: Performance of the GP with different cooperation mechanisms.

for each input value (averaging). The second method took the median member's output for each input value (median voting). For example, if at the first test point (x = -6.28) three team members produced outputs of: -0.8, 0.02, 121.4 the value used for calculating the error would be 0.02, the median value.

In the third method the input range is divided into N subranges and each team member is assigned a distinct subrange (preset ranges). Thus, with a five member team and 40 test points each member actually only has to fit 8 test points.

As with the team choice mechanism, each of the cooperation mechanism should favor either homogeneous or heterogeneous teams. It is hypothesized that averaging will favor homogeneous teams, as presumably each member must be reasonably close to the test function. The other two methods are expected to favor heterogeneous teams. Each member can specialize on a different range of the test function and outside of those ranges the functions are relatively free to vary. Thus, there does not seem to be any pressure to produce homogeneous teams.

4 Results

Figure 1 shows the error of the evolved functions over 100 generations for five test cases (the team size used for these trials is 5): heterogeneous teams with median voting, heterogeneous teams with preset ranges, heterogeneous teams with averaging, homogeneous teams (with median voting), and a normal GP trial (i.e. team size of 1). Each set of results is the average of fifty trials.

Figure 1 shows that median voting and preset ranges produce the best results. Averaging is roughly equivalent to simply evolving individuals, and the homogeneous teams perform worse than evolved individuals. Thus, it is clear that simply using teams is not a guarantee of improved results. The mechanisms for chosing the team and for cooperation are fundamental.

Table 2 summarizes the errors in the final generation for different team sizes. These data emphasize several facts. It is clear that median voting and preset ranges are beneficial and that for these cooperation mechanisms larger teams further improve the results, although the advantage levels off fairly rapidly. In contrast averaging the team results is actually harmful and increasing the team size degrades performance.

These results seem to support the hypothesis that cooperation mechanisms favoring heterogeneous and more specialized members (in this case median voting and preset ranges) produce better results. However, it is worth examining the actual solutions being created by the GP. The following figures show the best team after 100 generations taken from the fiftieth trial. It is entirely possibly that trial fifty produced a relatively

Table 2: Summary of team results. These are the errors for the best programs in the final generation, averaged across 50 trials. The last row shows the results evolving individuals, so the team size does not apply.

Team Type	3 Member Team	Std. Dev.	5 Member Team	Std. Dev.
Median voting	1.53	0.28	1.22	0.23
Preset ranges			1.49	0.25
Averaging	2.62	0.41	2.79	0.49
Homogeneous	3.17	0.68	3.20	0.77
Individual	2.77	0.55		

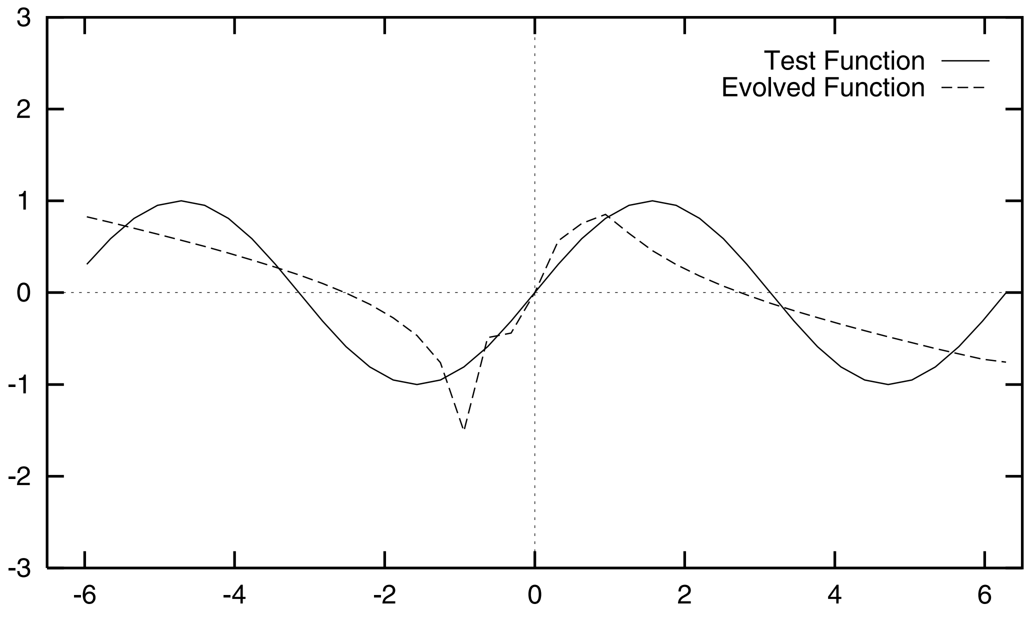

Figure 2: Solution generated when evolving individuals. Note that the evolved function corresponds reasonably well with the target function.

poor best individual. Thus, these figures are only considered for the general form of the solution, not its absolute fitness. For the team trials, the results are for teams with five members.

Figure 2 shows the best solution generated by a single individual. The generated function corresponds relatively closely to the test function, although this is clearly not a great solution.

Figure 3 shows the best solution generated by a team when each member assigned a preset range. The combined function is shown along with the function produced by 3 of the 5 team members. Within their assigned ranges each of the members generated a function that corresponds relatively closely to the test function. For example, member 3 is fairly accurate in its range (roughly -1.3 to 1.3). However, outside their range each members' function is very different from the test function. This is clearly cooperation, the members only perform well as a team. It is also clearly specialization, each member has evolved to be correct only within its own narrow range of specialization.

Figure 4 shows the best solution generated by a team using median voting. Again the combined function is shown along with the functions produced by 3 of the 5 team members. As in the previous case the combined function is relatively close to the test function, but the individual member functions are not. Here the team members have evolved regions of specialization within which their performance is good.

In this case several members have multiple, disjoint ranges of specialization. For example, member 1 fits the test function from -3.5 to -2.5 and again from 2.5 to 3.5. This requires a higher level of cooperation than seen in the previous example. Here the members must

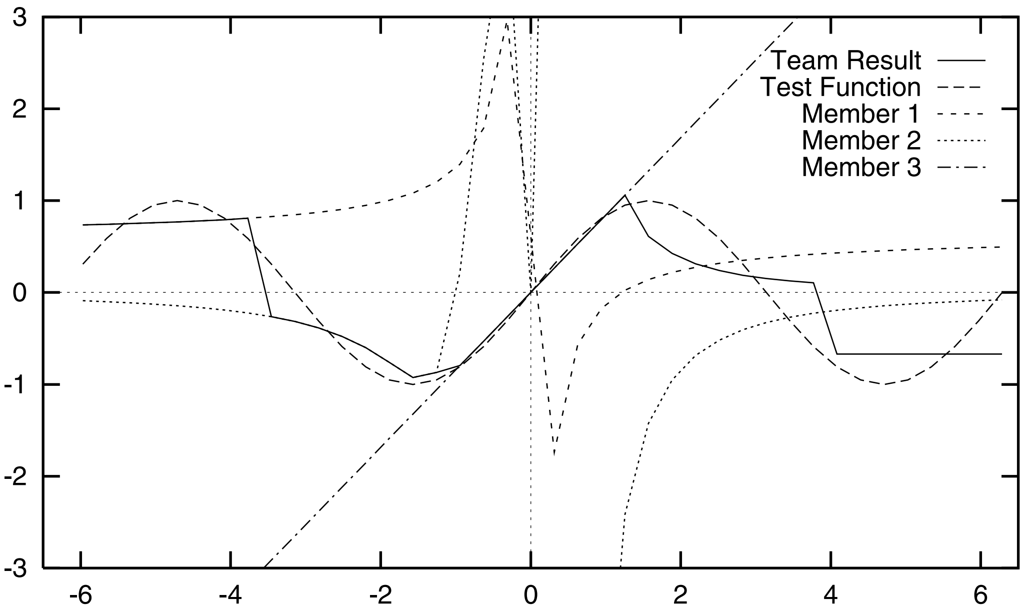

Figure 3: Solution generated by the team when using preset ranges of specialization. The individual member solutions are shown along with the combined result.

not only be correct within their ranges of specialization, they must also assure that the median vote applies to the correct other member of the team. Thus, it is clear that both specialization and cooperation are evolving.

The primary difference between these two examples is that with preset ranges each function is forced to fit a predetermined range whereas with median voting the ranges of specialization can evolve. For this problem allowing the ranges of specialization to evolve is somewhat advantageous. Presumably, because it allows the test function to be subdivided into fairly "natural" ranges. It seems likely that a conscious attempt by the programmer to choose preset regions would improve the results. However, for many problems this will not be possible. This is particularly true of problems were the ranges cannot be so clearly graphed. For example, it is not obvious where the "natural" ranges of specialization are in a data mining problem.

Figure 5 shows the results for teams using averaging as the cooperation mechanism. Here the results differ considerably from what was expected. We had expected averaging to result in fairly homogeneous teams in which each member was fairly close to the test function. Clearly this is not the result. The members are extremely diverse and do not even remotely resemble the test function. Despite this the average of the functions is relatively close to the test function. (Overall averaging produced solutions as good as produced by individuals.) Thus, the averaging cooperation mechanism clearly produces cooperation (as do the other two mechanisms), but does not lead to the homogeneous teams that were expected.

In addition, averaging does not produce the specialization that was seen with median voting and preset ranges. For any given range of the test function none of the members would do well by itself. This is in marked contrast to the previous team cases.

The results for homogeneous teams were closer to what was expected. In some cases the population converged entirely. Thus, median voting (or any other cooperation mechanism) had no real effect because the members were identical. In other cases the population did not converge, but the process of appling median voting to five individuals randomly draw from the population worked very poorly.

4.1 Generalizing Ability of the Solutions

How well the solutions generalize outside of the training data is another important consideration with machine learning techniques. With the linear regression problem generalization is further subdivided into two distinct cases: generalizing within the training range ($-2PI$ to $2PI$ in this case) and generalizing outside the training range ($< -2PI$ and $> 2PI$ in this case).

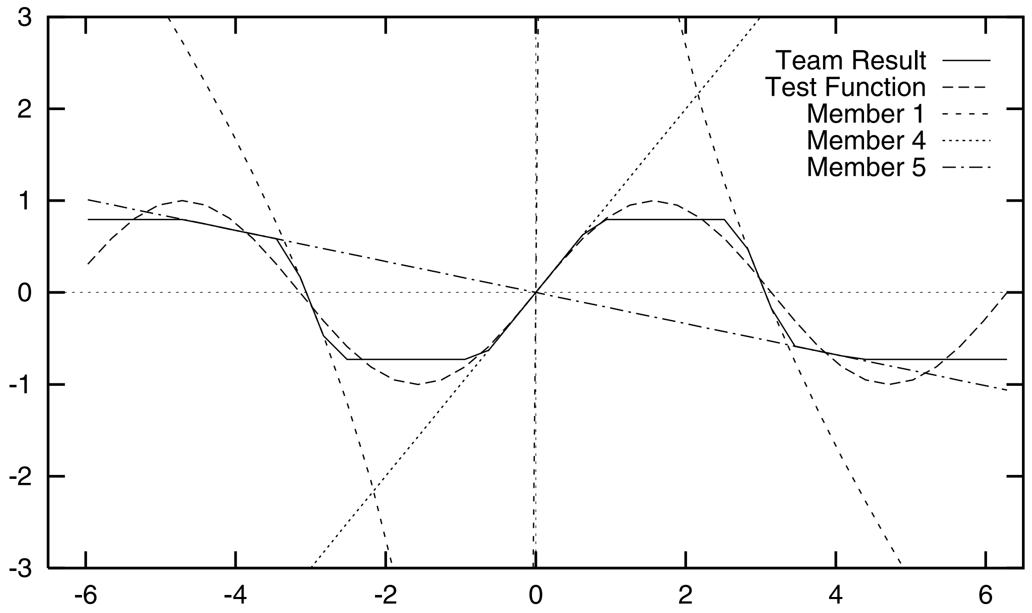

Figure 4: Solution generated by the team using median voting. The individual member solutions are shown along with the combined result. Although the team members are only accurate within narrow ranges of specialization, the overall result is fairly accurate.

The results are fairly clear from looking at Figures 2 through 5. Generalization within the training range is fairly good for all cases. The teams with more specialization give better results simply because they produced the more accurate functions.

In contrast, outside the training range the specializing teams will perform very poorly. This should not be surprising. With specialization each member's ability is limited to its range. Outside of the training range none of the members can be expected to do well. Thus, a major drawback of the teams with specializing members is that they will generalize poorly outside of the training range.

5 Conclusions

In these experiments team performance was dependent on two factors, team heterogeneity and specialization. These are clearly related. If a team consists of homogeneous members specialization is not possible. However, even when the team members are heterogeneous specialization is not guaranteed to occur. This was shown in the trials using averaging. Figure 5 clearly shows heterogeneous team members, but there does not appear to be any specialization among those members. In contrast Figures 3 and 4 show clear specialization, each member is accurate only for a subset of the training data.

Finally we saw that of these two factors (amount of heterogeneity and amount of specialization) the amount of specialization is key to improving performance. Heterogeneity is a necessary, but not sufficient, condition for specialization to occur. The team choice mechanism is important in producing heterogeneous teams. However, these experiments suggest that it is the cooperation mechanism that most favors specialization. Thus, developing an appropriate cooperation mechanism will be critical in applying teams to other problems. These experiments suggest that the mechanism should favor the evolution of independent areas of specialization.

There is a drawback to specialization in these experiments. When each of the team members specializes on a limited region of the training data the ability to generalize outside of the training range is degraded, although performance within the training range improves.

Finally, it is worth noting again that cooperation did evolve whenever heterogeneous teams were used. Thus, it seems reasonable to claim that cooperation is a fairly easy trait to evolve. This is a very significant result. It suggests that a team approach can be successfully applied to most problems.

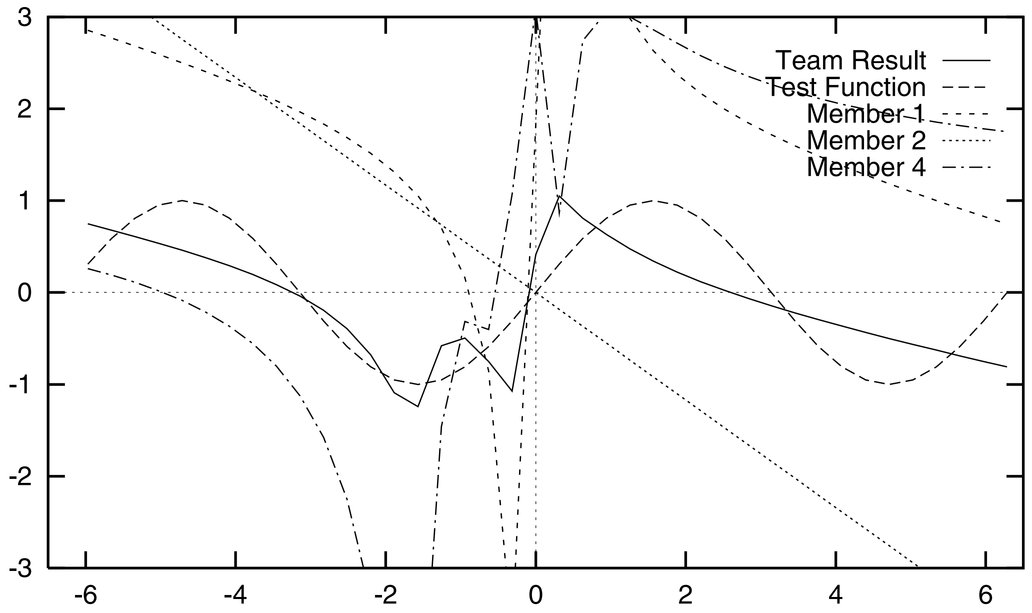

Figure 5: Solution generated by the team using averaging. The individual member solutions are shown along with the combined result. Interestingly the individual members are still quite poor and the average result is much better than any single member's result.

References

[HS97] Kim Harries and Peter Smith. Exploring alternative operators and search strategies in genetic programming. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick R. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 147–155. San Francisco, CA: Morgan Kaufmann, 1997.

[HSSW95] Thomas Haynes, Sandip Sen, Dale Schoenefeld, and Roger Wainwright. Evolving a team. In Eric V. Siegel and John Koza, editors, *Working Notes of the AAAI-95 Fall Symposium on GP*, pages 23–30. AAAI Press, 1995.

[Iba97] Hitoshi Iba. Multiple-agent learning for a robot navigation task by genetic programming. In John R. Koza, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick R. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 195–200. San Francisco, CA: Morgan Kaufmann, 1997.

[LS96] Sean Luke and Lee Spector. Evolving teamwork and coordination with genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick R. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference on Genetic Programming*, pages 150–156. Cambridge, MA: MIT Press, 1996.

[RD94] Simon Raik and Bohdan Durnota. The evolution of sporting strategies. In Russel J. Stonier and Xing Huo Yu, editors, *Complex Systems: Mechanisms of Adaption*, pages 85–92. IOS Press, 1994.

[Sou99] Terence Soule. Voting teams: A cooperative approach to non-typical problems. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 916–922, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.