Voting Teams: A cooperative approach to non-typical problems using genetic programming

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

Abstract

This paper presents a general approach for evolving cooperative teams to solve problems that are non-typical of team research. Cooperation is achieved by allowing the team members to vote on the solution. In these experiments the even-7-parity problem is used as the test problem. Although the even-7-pariety problem is not typical of team learning, cooperating teams outperformed individuals and used fewer resources. Interestingly, the team members' individual performance is quite poor; the improved performance is only achieved through cooperative voting.

1 Introduction - Teams and Voting

There are many examples of cooperative behavior in nature, such as pack hunting by wolves and lions or the use of sentries by geese and prairie dogs. It is generally assumed that the advantages (increased hunting success or reduced predation) of these strategies for the pack or flock outweigh the disadvantages (having to share kills or increased danger for the sentry). Recently several researchers have used genetic programming (GP) to evolve similar cooperative behavior between programs [HSSW95, LS96, HS97]. Most of the research in evolving teams has focused on problems requiring team solutions, particularly predator-prey pursuit games. The teams produced during this research were fairly successful and some general conclusions can be drawn from the experiments. In general, GP seems to have more success with heterogeneous teams, in which each member behaves differently, than with homogeneous teams [LS96, HS97]. Crossover and selection operations that encouraged heterogeneity generally performed better than operations that reduced heterogeneity. Luke and Spector showed that predator teams preformed better when the predators were able to locate each other by name (i.e. lion 1, lion 2 versus nearest lion, leftmost lion) [LS96]. This reinforces the argument that heterogeneity, increased by supplying each agent with a unique name, is important.

In related work Iba examined voting between the individuals of a population [Iba97]. The goal of his research was to take advantage of the "species knowledge" incorporated in the population as a whole. Any single individual, even the best individual of the population, is likely to be incorrect in some cases. However, there is a strong possibility that another individual in the population will code the correct answer for those cases, even though it is wrong on others. By using a weighted vote among the best individuals from the population Iba was able to significantly improve the GP's performance [Iba97].

2 Experimental Design

This research combines the team concept used by Haynes et al. and by Luke and Spector with the voting paradigm used by Iba to demonstrate that cooperating teams can be successful even on very non-typical problems, in this case the even-7-parity problem [Koz92, GR97]. As in the research described above, the "individuals" within the population are actually a team consisting of several (in this case 5) separate members. Keeping the team members together throughout the evolutionary process appears to improve the evolution of cooperation because the same team members are always working together.

For each problem instance each of the five members generates a solution. The solution of the team as a whole is determined by a vote among the members. This can be viewed as a variation of automatically defined functions (ADFs) [Koz94] with each member resembling a separate ADF. Note that the use of vot-

Table 1: Summary of the even-7-parity problem.

Objective:	Determine the parity of 7 Boolean inputs.
Terminal set:	The 7 input values
Function set:	`AND`, `NAND`, `OR`, and `XOR`
Fitness:	Number of correctly categorized cases (out of 2^N)
Population size:	500
Crossover probability:	90 percent
Mutation probability:	0 percent
Selection:	stochastic remainder
Termination criteria:	maximum generation
Maximum generation:	125
Maximum size of trees:	none
Initial population:	grow, no size limits
Number of trials:	fifty

ing requires heterogeneous teams. All the members of a homogeneous team, such as created by "cloning" a single individual, would vote the same, thus rendering the vote pointless.

The even-N-parity problem is to find a boolean circuit that correctly determines the parity of N boolean inputs. This is generally considered one of the harder boolean circuit problems. For 7 inputs there are a total of 128 test cases to be classified.

For this problem each team member generates a single Boolean value on each test case. The team's answer for each test case is the majority vote. Weighed voting is not used. Note that this approach avoids the credit assignment problem, as no attempt is made to determine which team member contributed the most to the team's overall performance. The general GP parameters are given in Table 1. These were chosen for consistency with other GP experiments, not to optimize the teams' performance. In the initial population each team member is generated using random growth.

In order to increase the heterogeneity of the team members restricted crossover was used. In restricted crossover team member 1 is only allowed to cross with member 1 of other teams, member 2 with member 2 of other teams, etc [LS96]. Restricted crossover is believed to increase specialization within the team, thereby increasing heterogeneity, and has been found to be more successful than many other breeding strategies [LS96, HS97]. Only one of the five team members (chosen randomly) is actually subjected to crossover during each crossover operation.

Fitness is the number of correct choices (out of all 128). Size is used as a secondary criterion. If two teams have the same fitness the smaller one is ranked higher for selection. This is the only control on size. Mutation is not used.

3 Potential Benefits of Voting Teams

Normally teams are applied to problems where multiple agents have a clear utility. Often multiple agents are required to achieve a solution (such as in predator-prey tasks where the prey could easily escape a single predator). The even-7-parity problem is not this type of problem. Clearly a single individual can solve the even-parity problem and it is not normally presented as a candidate for team solutions.

However, there are two general strategies by which teams could improve their performance on the even-parity problem. First, the team members may improve, thus, increasing the overall success of the team. Second, the team members may successfully partition the problem so that their limited knowledge is exploited to the maximum possible benefit. As an illustration of the second case consider the even-3-parity problem, which has 3 inputs and 8 test cases. The success of an individual is determined by the number of cases that it gets correct. If the individual is correct on half of the test cases (4 out of 8) it is fifty percent successful.

Now consider a team consisting of five members in which each member is correct on half of the test cases. By carefully choosing which members get which test cases correct the, overall performance of the team can be considerably above fifty percent, see Table 2. Table 2 demonstrates that a team can achieve a higher success rate by careful allocation of the correct votes. No correct answer receives more than three correct votes and the number of incorrect votes on the wrong

Table 2: Illustration of one benefit of voting. Although each member is correct in only 4 of the 8 cases, careful allocation of the correct votes makes the entire team correct in 6 of the 8 cases.

Input	Answer	1	2	3	4	5	Vote
000	1	1	1	1	0	0	1
001	0	0	0	1	0	1	0
010	0	0	1	0	1	0	0
100	0	0	1	0	0	1	0
011	1	0	1	1	0	1	1
101	1	0	1	0	1	1	1
110	1	0	0	0	1	1	0
111	0	1	1	1	1	1	1
	#Correct	4	4	4	4	4	6

answers is maximized (e.g. on input 111 all five members vote incorrectly).

In addition to the potential improvements in performance teams may also improve the evolutionary characteristics of the GP. A team is much less brittle than a single individual. A dramatic change, such as from crossover, to an individual is likely to have an equally dramatic effect on that individual's performance. However, a similar change to one team member is much less likely to have as great an effect on the performance of the team as a whole because the voting process will tend to cover changes in a single member. Thus, a voting team has built in-fault tolerance. This is not a surprising conclusion, as redundant, voting systems are a standard approach to improving fault tolerance.

4 Results

Two basic cases were compared: teams of size five and teams of size one (i.e. individuals). The results shown are the average of 50 trials.

4.1 Fitness

Figure 1 compares the average and best fitnesses of individuals and teams. It is clear that the teams do outperform individuals. The teams produced better average and best performances. The difference in average fitness is significant in all generations after the first and the difference in best fitness is significant in all generations (Student's two-tailed test, $P < 0.01$). In two of the trials a team which successfully classified all 128 test cases was found.

Examining the method by which the teams improve their performance leads to some interesting conclusions. The last curve in Figure 1 shows the average performance of solitary team members. The average performance of a solitary team member is only a little better than random, much worse than the evolved individuals or the teams as a whole. In other words, although the teams perform well, the individual team members are relativly unintelligent. It appears that the teams succeed by partitioning the problem to maximize the members' limited knowledge, as was illustrated in Table 2.

Figure 2 shows the number of input cases in which 0, 1, 2, 3, 4, or 5 of the members vote correctly. Initially, the distribution is roughly 4 20 40 40 20 4. I.e. in 4 cases no members vote correctly, in 20 cases 1 member votes correctly, in 40 cases 2 members vote correctly, in 40 cases 3 members vote correctly, in 20 cases 4 members vote correctly, and in 4 cases all five members vote correctly. This is the expected distribution in a random Gaussian population.

(There are 2^5 or 32 possible voting permutations. Only one of these has all members voting correctly and one has no members voting correctly. Five permutations have four members voting correctly and five have one member voting correctly. Ten permutations have three members voting correctly and ten have two members voting correctly. Thus, the odds of having 0, 1, 2, 3, 4, and 5 members voting correctly are 1/32, 5/32, 10/32, 10/32, 5/32, and 1/32 respectively. With 128 trials this gives the distribution (4 20 40 40 20 4) shown above.)

Over time the number of cases in which three members vote correctly rises as expected. However, the other results are more interesting. The number of cases in which four or five members vote correctly declines, even though in these cases the team is correct and is rewarded. In fact, this is a more efficient way of utilizing the limited knowledge of the team members.

In this experiment having three correct votes produces the same effect as having four or five correct votes. Thus, if the number of correct votes is viewed as limited commodity, it is more efficient to reduce the number of correct votes to three per case. Then the other correct votes can be distributed over other input cases.

Similarly, the number of cases in which only two members vote correctly is dramatically reduced. In the initial generation there are twice as many cases with two correct votes as there are cases with one correct vote. However, by the final generation there are considerably fewer cases with two correct votes than with one correct vote. Again this cooresponds to moving to-

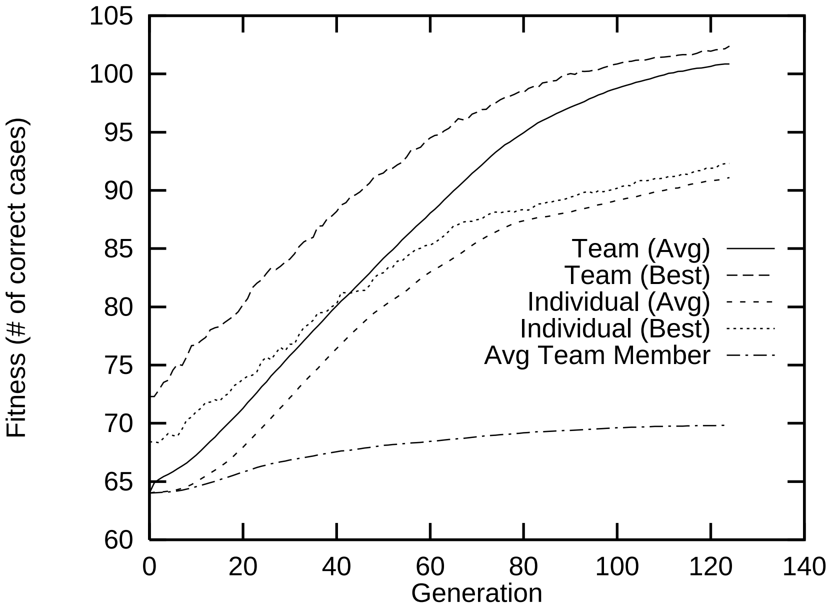

Figure 1: Performance of the GP with and without teams. The average performance of the individual team members (without voting) is also shown.

wards a more efficient case. Using two correct votes, while still getting the wrong answer is more wasteful of the correct votes than only using one correct vote.

In effect the GP has turned the even-7-parity problem into a distribution problem: how to distribute a limited resource (correct votes) to maximize the number of test cases with at least three correct votes.

This result can also be viewed positively in terms of building blocks. Each member of a team is effectively one of five large building blocks that contribute to the performance of the team as a whole. In this experiment the GP is clearly successful at selecting building blocks which complement each other.

There are several likely causes of the team members' poor performance. First, it is clear that the GP can perform quite well with unfit team members. Thus, there may be little incentive to improve them. Presumably, the fitness function contributes to this cause by not rewarding better team members over poorer team members.

Second, for the team members the crossover rate is actually quite low. Although there is a 0.9 chance of a team being selected for crossover, only one member of that team will undergo crossover. Thus, the effective crossover rate for a team member is only 0.18. This certainly contributes to the members' slow improvement.

4.2 Code Size

Figure 3 shows the average code size for individuals and teams. The code size for the teams is calculated by summing the size of all team members. Thus, in the initial population the average size of the teams is five times as large as the average size of the individuals. Interesting, this disparity in size does not last. By generation 23 the individuals contain more code than the teams. Then in the later generations the average size of the individuals again drops below the average size of the teams and the two sizes remain relatively close together. The sum of the program size over all 125 generations is, teams: 28,470 program nodes in an average run, individuals: 29,980 programs nodes in an average run. Thus, over the course of all 125 generations the average team size is slightly smaller than the average individual size. This means that overall slightly less resources were required for the team trials, even though initially the teams were five times as large as the individuals. This was an unexpected

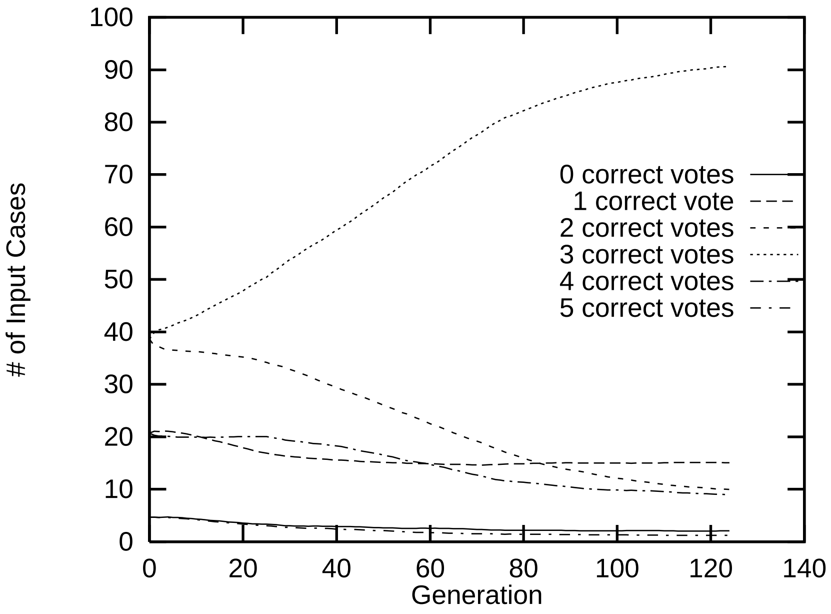

Figure 2: Number of cases in which 0, 1, 2, 3, 4, or 5 members voted correctly.

benefit of the team approach.

There are two likely reasons for the reduced growth with teams. First, there is strong evidence that growth is a protective mechanism against the destructive effects of crossover [NB95, SF97]. Extraneous code appears to shelter important code from the normally destructive effects of crossover. If the ratio of important code to extranious code is small, a crossover at randomly chosen site is unlikely to effect important code, making it less likely to decrease the program fitness. However, if a member of a team is damaged during crossover the votes of the other members will minimize the negative effect. Thus, the potential damage from crossover is reduced. This reduces the evolutionary advantage of protective code and may lower the evolutionary pressure towards code growth. In effect the inherent fault tolerance of voting teams reduces the destructive effects of crossover and reduces the need for extraneous growth.

Second, with teams the variations in size caused by crossover will be smaller, which will reduce the opportunities for rapid growth. On average each crossover/selection combination causes a slight increase in the size of the effected program. Individual team members grow more slowly because they are crossed only one fifth as often as individuals. This means that the candidates for crossover with teams will be (after several generations of growth) smaller than the candidates for crossover without teams, even though the overall size of the team may be larger. Because the candidates are smaller, the size variation produced by crossover will be smaller and the overall growth rate will be smaller. This effect also contributes to the smaller size fluctuations demonstrated by the teams.

4.3 Optimal Parameters

In these experiments no attempt was made to optimize the parameters for the teams. For example, the crossover rate of 0.9 was chosen because it is commonly used and is assumed to be near optimal for normal, non-team, GP. Table 3 shows the average fitness at generation 125 when different crossover rates were used. It is clear from these results that a higher crossover rate (or possibly crossing more than one team member at a time) would have benefited the teams' evolution.

The choice of team size for these experiments was relatively arbitrary. Clearly an odd size is beneficial for easy voting. (Although an even team size with ties treated as "don't know" and given a distinct fitness is an interesting possibility for future research.) However, in these experiments success was achieved by distributing the correct votes. Thus, it seems likely that

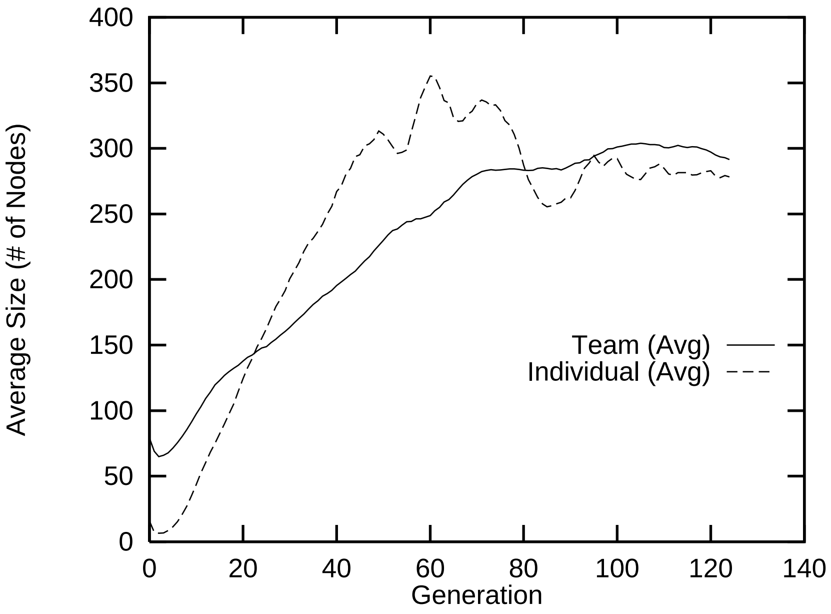

Figure 3: Average size of the evolved indivdiuals. Team size is calculated by summing the sizes of all five members.

Table 3: Effect on team performance of different crossover rates.

Crossover Rate	Avg. Team Fitness (Generation: 125)
0.8	99.0
0.9	101.0
1.0	102.0

larger team sizes would be beneficial because larger teams would permit a more balanced distribution of correct votes. Future research will be devoted to finding better parameters for voting teams.

5 Conclusions

This research demonstrates that it is possible to evolve teams to solve problems that are not traditional team type problems. Cooperation is achieved through voting. The evolved teams are both more effective and more efficient (smaller) than the evolved individuals. The relative lack of growth with the teams suggests that the team appoach also produces some inherient fault tolerance.

Perhaps more importantly, it was demonstrated that for these problems GP evolves *cooperating* teams. The success shown here was based on evolving cooperation between relatively poor team members, not on evolving knowledgeable team members. This suggests that even a very limited form of credit assignment could significantly improve the results. If the individual team members evolved slightly more rapidly, the overall improvement after voting should be dramatic.

6 Future Work

As noted previously, no attempt was made to optimize the parameters for the team approach. Significant improvements are expected when more appropriate parameters are used. Similarly several team oriented crossover operations have been presented in the literature [HS97]. The one used for these experiments seemed most suited to this task, however it is possible that more appropriate versions of crossover exist.

Finally, no attempt was made to study the fault tolerant features of the teams. The reduced growth rates suggest that the teams do have some inherent fault tolerant properties, which limit the need for excess growth. Clearly such properties would be extremely beneficial in practical applications and should be explored further.

References

[GR97] Chris Gathercole and Peter Ross. Tackling the boolean even n parity problem with genetic programming and limited-error fitness. 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 119–127. San Francisco, CA: Morgan Kaufmann, 1997.

[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.

[Koz92] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: The MIT Press, 1992.

[Koz94] John R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. Cambridge, MA: The MIT Press, 1994.

[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*, 1996.

[NB95] Peter Nordin and Wolfgang Banzhaf. Complexity compression and evolution. In Larry J. Eshelman, editor, *Proceedings of the Sixth International Conference on Genetic Algorithms*, pages 310–317. San Francisco, CA: Morgan Kaufmann, 1995.

[SF97] Terence Soule and James A. Foster. Code size and depth flows 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 313–320. San Francisco, CA: Morgan Kaufmann, 1997.