Orthogonal Evolution of Teams: A Class of Algorithms for Evolving Teams with Inversely Correlated Errors*

Terence Soule[1] and Pavankumarreddy Komireddy

Department of Computer Science, University of Idaho, Moscow, ID, 83844-1010
tsoule@cs.uidaho.edu

Summary. Several general evolutionary approaches have proven quite successful at evolving teams (or ensembles) consisting of cooperating team members. However, in this paper we demonstrate that the existing approaches have subtle, but significant, weaknesses. We then present a novel class of evolutionary algorithms (orthogonal evolution of teams (OET)) for evolving teams that overcomes these weaknesses. Specifically it is shown that a typical algorithm from the OET class of algorithms successfully generates team members that have fitnesses comparable to those evolved independently and that have inversely correlated errors, which maximizes the teams' overall performance. Finally it is shown that the OET approach performs significantly better than the standard evolutionary approaches.

1 Introduction

Many real-world problems are too large and too complex to expect a single, monolithic intelligent agent to solve them successfully. Monolithic agents, because they are monolithic, tend to overlook specialized sub-domains within a larger problem space and thus to make errors on those sub-domains. As programs are expected to solve progressively larger and more complex problems this weakness will become increasingly critical. Thus, considerable research has been devoted to developing problem solvers consisting of integrated subsystems *specialized* to find solutions within sub-domains of the total problem space. These structured solutions are robust and can solve problems defined over huge domains with no (or very few) gaps or errors. Much of this research has focused on team, or ensemble, learning in which each member of the team receives the same inputs and solutions are determined via a cooperation mechanism, such as a vote. Such cooperative teams have been successful at solving complex problems ranging from predicting disordered regions in proteins [21] to learning musical models [28] to weather prediction [19].

* This research supported by NSF 0535130

There are two fundamental requirements to creating successful teams. The individual team members must be relatively successful and the team members must cooperate in a way that improves the performance of the team as a whole. Typically this means that the members *specialize* on distinct, but potentially overlapping, sub-domains of the problem space. To be deemed successful any technique should be shown to meet both of these requirements.

Recently evolutionary techniques, including both genetic algorithms (GAs) and genetic programming (GP), have proven to be extremely effective at automatically *evolving* teams. Evolutionary approaches for creating teams have been applied to a wide range of knowledge representations including teams of: neural networks[16], oblique decision trees [4], stack based predictors [22], and teams of induced functions [25] and to a wide range of problem domains including robot navigation [9], sporting strategies [23], predator strategies [7, 17], hazard assessment [20], and cancer and diabetes diagnosis [4, 16]. These results demonstrate the effectiveness and broad applicability of evolutionary techniques to the problem of automatically generating teams.

However research suggests that the two general approaches, *island* and *team*, each have significant, if subtle, weaknesses. Island approaches produce highly fit team members, but there is a high probability that those members have *correlated* errors resulting in less than optimal team performance [11, 12]. In contrast team approaches can produce team members with *inversely correlated* errors leading to relatively good team performance, but the members themselves are relatively poor, which limits the teams' performance [27].

In this paper we use a simple illustrative problem and the standard benchmark problem inter-twined spirals to confirm the weaknesses of the island and team approaches and introduce a novel evolutionary algorithm design: Orthogonal Evolution of Teams (OET) that overcomes these weaknesses. OET defines a new class of cooperative, co-evolutionary algorithms that evolves team members with distinct areas of specialization. Our results show that OET algorithms combine the advantages of both the island and team approaches. OET algorithms generate highly successful individual solutions, whose members specialize on distinct sub-domains of the problem space, leading to robust solutions that cover an entire problem domain with no (or few) gaps or errors.

2 Background

In this work specialization within a team is directly measured using the idea of error correlation between team members. If members have evolved to specialize on distinct (but over-lapping) sub-domains of the problem space then their errors will be *inversely* correlated. Correlation is measured using the expected failure rate model that was initially derived for the investigation fault tolerant systems.

Fault tolerance is the ability of a system to perform despite faults or deficiencies within components of the system. It is fundamental to ensuring the

reliability, availability and accuracy of a system and is a requirement for critical systems. Commonly software systems with redundant components achieve fault tolerance via a majority vote. For example, if five redundant components are used ($n = 5$) then three of the components ($m = 3$) must fail concurrently to produce a failure.

For fault masking to successfully mask errors (at least in software) the redundant members must be different. If one member has an error that causes the member to exhibit a fault on a particular input then making n copies of the member and having them all vote will not solve the problem. Thus, for fault masking to be successful the redundant components, e.g. team members, must be significantly different from each other. The degree of difference is measured by calculating the correlation of errors between team members. If errors are highly correlated the team members will tend to return errors for the same input cases and there is little advantage to forming a team. If the errors are independent, or better, are inversely correlated, then the members will rarely return errors for the same input cases and the performance of the team will be dramatically better than the members' performance. This can be viewed as a measure of cooperation - the team performs much better than the individual members. Or as a measure of specialization - inversely correlated errors imply that the members have specialized to cover distinct sub-domains of the problem space.

To determine whether the errors between voting team members are correlated, independent or inversely correlated we use the expected failure rate model [1]. Let n be the number of team members, let p be the probability of each member producing an error for a given input and let m be the minimum number of concurrent errors required for a system to fail. The expected failure rate (f) for the whole system, assuming independent errors between the team members, is

$$f = \sum_{k=m}^{n} \binom{n}{k} (1-p)^{(n-k)} p^k \tag{1}$$

If the failure rate of a system with redundant components is greater than f then the errors are correlated. If the failure rate is equal to f then the errors are uncorrelated. This is generally assumed to be the best achievable condition for fault tolerance. However, if the failure rate of the system is less than f then the errors are inversely correlated. This can only occur if the team members are specifically designed to have this property. Any approach that produces the members independently can, at best, be expected to produce independent faults.

In N-version programming, which is commonly used to produce fault tolerant software for critical computing systems, independent programming teams, often from independent companies, produce software from the same set of requirements [1, 8]. The software produced by each programming team is used as one of the redundant components. N-version programming produces

software components that individually have very few faults, as few as can be achieved via hand coding. However, studies of N-version programming by Knight and Levenson [13] and by Hatton [6] have shown that N-version programming still produces software components with correlated faults, even when independent design and coding methodologies are used. They hypothesized that this occurs for several reasons: human programmers tend to make the same (incorrect) assumptions and the same logic errors, and most programmers learned similar design and coding strategies.

To avoid the problems introduced by human programmers a number of researchers have examined using evolutionary computation (EC) to evolve teams. Because EC is a highly randomized process it is assumed that any errors will be randomly distributed. In fact, this assumption is suspect. EC generally takes the path of least resistance; the pressure to survive encourages EC to find the easiest solutions first. Thus, it seems likely that even independent populations will tend to favor the same general solutions and may produce correlated errors.

N-Version Genetic Programming (NVGP) developed by Imamura et al. evolves n independent populations (islands), when evolution halts the NVGP algorithm draws a random individual from each population to create a team of n individuals [11, 12]. Although NVGP produces some teams with independent errors for a range problems, the majority of teams are *not* qualified. In general, it takes a large number of random draws to find a qualified team. This appears to invalidate the assumption that independent evolutionary runs will consistently generate solutions with independent errors.

Several other researchers have experimented with algorithms using independent populations, although without directly measuring predicted failure rates [5, 29]. In general, it seems likely that any approach that uses independent populations (island approaches) will have a similar flaw.

An alternative to the island approach is the team approach: in team approaches a single population evolves; each individual in the population represents a team of n components [7, 17, 25, 2]. The fitness of an 'individual' (which represents a team) is based on the performance of the entire team. Studies of the team approach strongly suggest that it avoids correlated errors, each team member evolves to have few errors on a particular domain of the problem and the domain for each member is different [25, 2]. Thus, the whole team has very few unmasked errors. Although none of the studies directly measured expected failure rates the results strongly suggest that the member errors are inversely correlated. This is a reasonable result as all of the selective pressure is on the team - there is direct pressure on the individuals to evolve teams consisting of members that mask each others' errors.

Unfortunately, team based evolution has its own drawbacks. When the team members are tested as isolated solutions they perform significantly worse than evolved individuals, even though the team performs better than those same individuals [26, 2]. Thus, the team approach produces highly fit teams

consisting of relatively poor individuals; whereas the island approach creates a reasonable team from highly fit individuals.

Bagging and boosting are two closely related techniques that have been combined with evolutionary techniques to create voting teams. Bagging uses a different set of training instances for each training round of the learning algorithm [3]. The training sets are created by randomly sampling, with replacement, from the set of all training instances. The team is created by combining the best individuals from each of the training iterations weighted according to how well the individual performs.

Boosting uses the entire set of training instances at each iteration of the learning algorithm. However, each training instance has a weight associated with it and the weight of the instances change with each iteration. Typically, training instances that were handled correctly in previous iterations have their weight reduced and training instances that were handled incorrectly have their weight increased [24].

In combination algorithms an individual from each training round is kept to create the team, as in the bagging algorithm, and the instances used in each iteration have different weights, as in the boosting algorithm. Typically, the instances chosen for a given training round are those with the highest current weights (i.e. the instances that have proven hardest to solve). In 1999, Iba used boosting and bagging with evolutionary computation [10]. His experiment validated that these techniques produce teams that perform better than individuals. There is evidence that boosting can introduce a large bias in the evolved teams, particularly when the training set is noisy or includes a few very difficult instances which will tend to dominate the training [24, 18, 14, 20]. This seems likely to produce members that are highly correlated because they have all evolved to focus on the same, heavily weighted, instances.

The algorithm proposed in this paper combines the island approach with the team approach to produce teams with highly fit components and with inversely correlated faults (or inversely correlated misclassifications for classification problems). We refer to our approach as orthogonal evolution of teams (OET).

3 Orthogonal Evolution of Teams

OET defines a class of algorithms designed to put evolutionary pressure on both the evolving teams as a whole (as in existing team approaches) and on the team members (as in existing island approaches) by alternately treating the evolving population as a series of N independent populations/islands and as a single population of teams of M members.

The basic OET algorithm follows the steady-state evolutionary model: two offspring are generated each iteration and replace two existing individuals in the population. The specific algorithm tested here is:

```
Generate a population of teams
Repeat for X iterations{
   Repeat for each of the N islands{
      Select two highly fit team members
      // This produces two new team of highly fit
      // members to act as parents.
   }
   Apply crossover and mutation
   // This generates two offspring teams
   Select two low fitness teams to delete
   Insert the two offspring teams
   }
}
```

Here team members must have high fitness to be selected to act as parents and teams must have a high fitness to avoid being selected for replacement. Several similar algorithms could be employed. For example, teams could be selected for crossover and members for replacement or whole teams and individual members could be selected on alternate iterations.

To test the effectiveness of OET we test the following hypotheses:

1. OET produces team members whose fitness is significantly better than members produced with the team approach.
2. OET produces teams whose errors are inversely correlated. This shows that OET produces members that cooperate better than members evolved using island approaches.
3. OET produces teams whose performance is significantly better than evolved individuals.
4. OET produces teams whose performance is significantly better than teams evolved using either island or team approaches.

4 Experiments

Two problems are used to test these hypotheses: a simple, illustrative problem and a standard benchmark problem, the inter-twined spirals problem [15]. Two versions of the illustrative problem are used, an unbiased version and a biased version, described in detail below.

4.1 Illustrative test problem

The illustrative test problem was specifically developed so that it would be easy to analyze the performance of the algorithms, particularly with respect to their error rates. In this problem there are 100 abstract 'errors', numbered 1 through 100, that can occur. The evolved members are integer vectors consisting of 70 integers in the range 1 to 100. Each integer represents a successfully

detected or avoided error. For example, the individual member represented by the vector:

5|15|3|5|2|12|15

avoids errors 2, 3, 5, 12, and 15 and would have a fitness of 5 (a success rate of 5%) because it 'avoided' five of 100 the potential errors. Because individual members are only 70 integers long they can at best avoid 70 of the 100 errors. Having the same integer repeated (e.g. 5 and 15 in the above example) confers no additional benefit.

This simplified problem has several significant advantages for analyzing team behavior. First, the number of errors is known. Second, we can determine which errors are avoided by each individual. Third, the difficulty of avoiding a particular error is the same for every error, but can be changed (see the biased problem below). Finally, a perfect solution requires a team; any single member must have at least a 30% failure rate.

Table 1. Summary of the evolutionary algorithm parameters for the illustrative problem (middle column) and inter-twined spirals problem (rightmost column).

Fitness	Number of unique integers	error rate
Integer values	1 through 100	NA
Function Set	NA	+, -, *, /, sin, cos, ifIt
Terminal Set	NA	X, Y, Constants
Population Size	500	400
Mutation Probability	0.014	0.01
Selection	3 member tournament	
Run Time	500 Generations	200,000 iterations 600,000 for non-teams
Initial Population	Random individuals of length 70	ramped half and half
Number of trials	100	40

Table 2. Expected failure rates for majority vote teams with 3, 5, and 7 members each of which has a 0.3 failure rate and independent faults.

Team Size	Expected Failure Rate	Expected Success Rate
3	21.60	78.4
5	16.31	83.69
7	12.60	87.4

Voting

For teams fitness is measured via majority vote in which each agent gets only one vote per fault. For example, given the three members:

5|15|3|5|2|12|15
2|11|9|7|2|15|21
1|17|3|9|1|17|17

errors 2, 3, 9, and 15 are avoided. Note that the errors do not have to have the same position in two team members to be masked by the vote. E.g. in the above example error 9 is avoided because members 2 and 3 both list it, at positions 3 and 4 respectively. However, the errors are only counted once per member. E.g. error 17 is not masked because it only appears in member 3, albeit 3 times. Other approaches are clearly possible, but this one most closely resembles the voting process commonly used in real world problems.

A simple and a biased version of this problem is used in these experiments. In the simple version every integer is equally likely in the initial, random population. None of the errors are easier or more likely to be found.

In the biased version of the problem the initial random individuals consists of integers in the range 1 to 80 only. The values 81 through 100 can only be obtained through mutation. This is analogous to a problem where some cases are relatively easy to solve (here cases 1 through 80) and some cases require additional searching (here 81 through 100).

We hypothesize that the island approach will produce correlated faults with the biased problem because in the island model there is no interaction between the islands to encourage independence. We further hypothesize that both the team approach and OET will generate independent or better members even with the biased problem.

Test algorithms

Three algorithms, each based on a steady-state algorithm, are compared. The first algorithm is an island approach: n islands are allowed to evolve and the best individual from each island is used to form a team. The second algorithm uses the team approach: one population is evolved and each individual in the population is a team. The final algorithm is the OET algorithm described previously. Note that the results of single individual from a single in the island approach is equal to what would be achieved using a (steady-state) non-team approach.

Crossover is performed by randomly choosing one member of the first parent team and performing two point crossover with the equivalent member from the second parent team. Thus, member X of parent one is always crossed with member X of the second parent. This method has been show to improve the evolution of specialization and overall team performance [7]. The parameters for the steady-state algorithm are given in Table 1. Mutation changes an integer to a randomly selected integer in the range 1 to 100. Each of the three algorithms is tested with teams of size 3, 5 and 7 on both the biased and unbiased problems. Each test is run 100 times.

Computational complexity

A non-team, steady-state algorithm is used as the baseline for measuring the computational complexity of these algorithms. It requires two evaluations per iteration, one evaluation for each of the offspring. So, n iterations requires $2n$ evaluations.

The island approach evolves m independent populations, each of which uses a non-team, steady-state algorithm. Thus, the island approach requires $2nm$ evaluations. The team approach uses a single population so it requires $2n$ evaluations, but evaluating an 'individual' requires evaluating m team members. Thus, the total number of evaluations is also $2nm$. The OET algorithm also requires a total of $2nm$ evaluations, because it generates two teams for evaluation per iteration. We are assuming that the computational complexity of the voting process itself is negligible.

Expected Failure Rates

For each team the error rate of the members is measured using the expected error rate given by Equation 1. The expected error rates for teams of 3, 5, and 7 members whose individual failure rates are 30% are given in Table 2. A 30% failure rate is used because it represents the best possible performance by a single team member.

Results - illustrative problem

Figure 1 shows histograms for the six test cases (island, team, and OET approaches on the biased and unbiased problem). The histograms show the distribution of the members of the best teams and of the best teams as a function of fitness in the final generation. The data is plotted for the teams with three members so there are three sets of data for the team members totaling 300 members (100 trials * 3 members per best team per trial). The bars for the team members tend to overlap because the members all have similar fitnesses. The bars for the best team total 100 (the best team from each of 100 trials) and show that the teams consistently perform better than the team members.

The upper-left histogram shows the distribution of fitnesses for the island approach on the unbiased problem. As expected the individual members are clustered very closely to the 70% success rate, with 222 members out of 300 having the optimal success rate. The success of the teams created by combining these members are distributed around the 78% success rate with an average success rate of 78.31%. This is not significantly different from the success rate predicted by the expected failure rate model (78.4%) (Student's t-test, t = 0.28, P ¿ 0.01). (The results for the final generation for all of the algorithms and all three team sizes are summarized in Table 3.) Thus, the island approach produces members with independently correlated faults.

The upper-right histogram shows the distribution of fitnesses for the island approach on the biased problem. Again all three of the islands produce individuals with optimal or very close to optimal fitness. However, the average fitness of the best teams is now 77.13%. This is significantly below the 78.4% success rate predicted by the expected failure rate model (Student's t-teast, $P < 0.05$) and significantly below the results of the island approach on the unbiased problem. Thus, for problems in which some of the cases are easier to find solutions to the island approach does not produce members with independently correlated faults. If the bias were sufficiently extreme it is possible that the island model would produce individuals with perfectly correlated faults (each island would find the same solution) and there would be no benefit from producing teams.

The middle two histograms show the distribution of fitnesses for the team approach. For both the biased and unbiased problem the team members are distributed around a success rate that is lower than the optimal. E.g. for the biased problem the average member fitness is 67%, which is slightly, but significantly, poorer than those produced with the island approach (Student's two-tailed t-test, $t = 16.2$, $P < 0.01$). However, the average team success rates are quite good 98.08% and 97.39%; these significantly exceed the performance of the island approach ($t = 82.9$, $P < 0.01$ for the unbiased problem) and the predicted failure rate. Thus, confirming that the team approach does produce members with inversely correlated errors. The results also confirm that the team approach produces poorer members than the island approach. For this problem the difference is relatively small (see Table 3), but for more difficult problem the difference is more significant (see the inter-twined spirals problem below).

The last two histograms show the distribution of fitnesses for the OET approach. These histograms make it clear that the OET approach does combine the advantages of the island and team approaches. The team members are near optimal (average success rates of 69.89% and 69.84%). Further, the performance of the teams generated with the OET approach clearly exceeds the 78.4% success rate predicted by the failure rate model. This is significantly better than the team approach (e.g. $t = 22.3$, $P < 0.01$ for the unbiased problem). Thus, the OET approach clearly produces highly fit members whose errors are inversely correlated.

Table 3 summarizes the results in the final generation for all three team sizes on both the biased and unbiased problems. The results for teams of size 5 and 7 are similar to those observed for teams of size 3. The island approach produces optimal or near optimal individuals and teams whose fitnesses match the prediction of the expected failure rate model for the unbiased problem, but not for the biased problem. Confirming that the island approach is likely to fail to produce members with uncorrelated errors if some instances are easier to solve. Interestingly as the team size increases this becomes less of a problem.

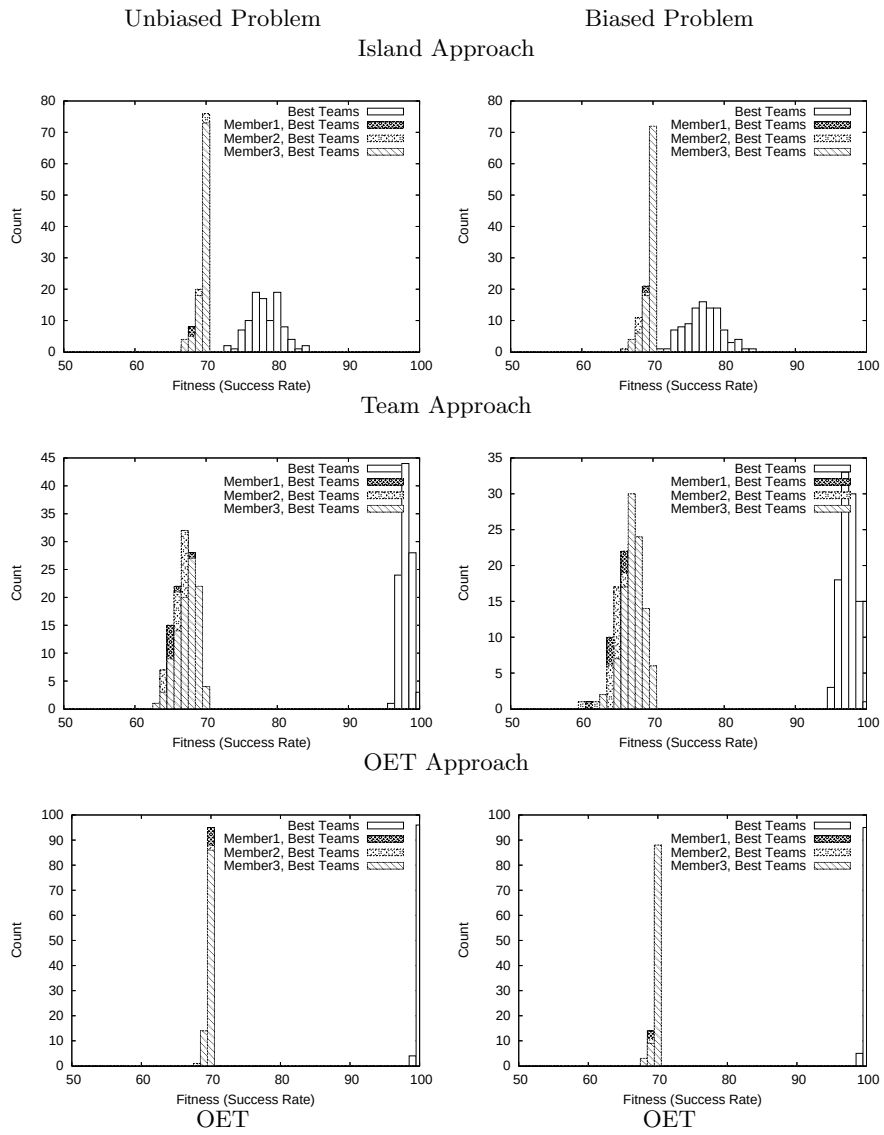

Fig. 1. Distribution of fitnesses for each of the three algorithms (island, team, and OET top-to-bottom) on the unbiased (left column) and biased (right column) problems. Bars for the team members overlap significantly.

For all team sizes the team approach produces sub-optimal individuals, but teams whose fitness significantly exceeds the fitness predicted by the expected failure rate model. Importantly, Table 3 shows that for the team approach, as the team size increases the average fitness of the team members declines signif-

icantly, resulting in poorer overall performance. This is a significant weakness of the team approach. We believe that the decline in member performance occurs because in the team approach selection is based on the team's fitness; the individual team members are only indirectly subjected to selection pressure and poor members may 'hitchhike' along with better members. As the teams become larger the importance of each member is decreased and the hitchhiking effect may be emphasized.

Table 3. Average results for the best teams and members of the best teams for the biased and unbiased problems. Values in parentheses are standard deviations.

		Unbiased		Biased	
		Best Teams	Members of Best	Best Teams	Members of Best
Size 3	Island	78.31(±2.24)	69.64(±0.69)	77.13(±2.58)	69.53(±0.81)
	Team	98.08(±0.82)	67.01(±1.47)	97.39(±1.07)	66.79(±1.64)
	OET	99.96(±0.20)	69.89(±0.32)	99.95 ± 0.22)	69.84(±0.43)
Size 5	Island	83.51(±2.35)	69.68(±0.66)	77.54(±1.67)	69.76(±0.48)
	Team	97.57(±1.18)	64.50(±1.97)	92.85(±1.31)	63.11(±2.16)
	OET	100(±0)	69.56(±0.65)	100(±0)	69.48(±0.77)
Size 7	Island	86.93(±2.61)	69.56(±0.80)	85.47(±2.75)	69.62(±0.73)
	Team	93.45(±1.66)	62.59(±2.25)	87.35(±1.74)	60.35(±2.35)
	OET	100(±0)	69.25(±0.85)	99.97(±0.17)	69.24(±0.81)

In all cases the OET approach produces highly fit members and highly fit teams. Unlike with the team approach member performance does not decline as team size increases. Demonstrating that the selective pressure applied directly to the team members in the OET approach is sufficient to maintain the members' fitness.

4.2 Inter-twined Spirals

The next set of results is based on the inter-twined spirals problem. For the inter-twined spirals problem two spirals coil around the origin of the x-y plane. Each spiral is defined by 97 points along its length. The goal is to find a function that classifies the points as belonging to spiral 1 or spiral 2 based on the x,y value of the point. An evolved program return a real value. Values less than zero are mapped to spiral 1, values larger than or equal to zero are mapped to spiral 2.

The function set is {+, -, *, /, `iflte, sine, cosine`}. Division is protected as in the symbolic regression problem. The function `iflte` ("if less than else") takes four arguments, if the first is less than the second then the third argument is returned, otherwise the fourth argument is returned. The terminal set is {x, y, constant}; x and y are the x,y values of the point to be classified. Constants are real values, randomly generated in the range (-1.0,1.0). For constants, mutation adds a random value in the range (-1.0,1.0)

to the constant's current value. Fitness is the percent of points misclassified. 0 is best, no points misclassified; 1.0 is worst, all points misclassified. On average a random classification will receive a fitness of 0.5. Parameter details are givien in Table 1.

Four algorithms are tested for the inter-twined spirals problem: evolved individuals (normal GP), the island model (with 3 islands), the teams model (teams of size 3), and OET (teams of size 3). The island, team, and OET approaches require three times as many evaluations as evolved individuals for the same number of iterations. Thus, to fairly compare the algorithms the evolving individuals are allowed to evolve for three times as many steady state iterations (600,000 iterations versus 200,000).

Table 4. Results for the inter-twined spirals problem. Values in parentheses are standard deviations. The final column (predicted) is the predicted error rate of the average teams for the team and OET approaches and for the best teams (the only ones generated) with the island approach. For both the team and OET approach the actual failure rate is better (lower) than the predicted rate, showing that the members' errors are inversely correlated. For the island model the predicted error rate and actual error rate are similar showing that the errors are independent.

	Best Teams	Average Teams	Average Member	Predicted
Individuals	0.096(±0.077)	0.1158(±0.0780)	–	–
Team	0.0813(±0.0555)	0.1160(±0.0517)	0.3242(±0.0609)	0.2472
OET	0.0439(±0.0357)	0.0654(±0.0377)	0.1806(±0.0576)	0.0861
Island	0.0492(±0.0293)	–	0.1375(±0.0717)*best*	0.0515

Table 4 presents the results, including expected team fitnesses, where appropriate. The performance of the members generated using the team approach is very poor, however, the teams' performance exceeds the expected performance confirming that the team approach produces teams members with inversely correlated errors.

In contrast the performance of the members evolved with OET are fairly good, although they are not as fit as the evolved individuals. The performance of the best teams evolved with the OET approach is significantly better than either the evolved individuals (Student's t-test, $t = 4.74$, $P < 0.01$) or the evolved teams (Student's t-test, $t = 4.65$, $P < 0.01$). Comparison of the predicted and actual average team fitnesses confirms that OET also generates members with inversely correlated errors.

The members evolved with the island approach are also very fit (they are not as fit as the evolved individuals because the individuals evolved for 3 times as many iterations). The average fitness of the best teams evolved with the island approach is equivalent to the prediction of the expected failure rate model, confirming that the island approach produced members with independent failures.

5 Conclusions

The results confirm that island approaches do produce members with independent errors, but only if all possible solutions are equivalent. For problems where some solutions are easier to find (e.g. the biased problem) all of the islands are likely to find the same, easier solutions, leading to correlated errors and decreasing the advantage of using teams.

Second, the results confirm that team approaches do produce members with inversely correlated errors, but with relatively poor members. This agrees with previous results that suggested that team approaches produce relatively poor members, but teams with high degrees of specialization and cooperation. The results also suggest that the drawbacks of having poor members will increase as the team size increases (Table 3) a serious limitation on the ability of team approaches to scale.

Third, OET clearly combines the advantages of the island and team approaches. OET generated both highly fit members and generated teams with inversely correlated faults.

Most importantly the results show the importance of considering both the fitness of the members and of the resulting teams. In particular, how the teams' fitness compares to the expected fitness as predicted by the expected failure rate model. The goal of any team or ensemble algorithm should be to optimize both member performance and the degree of correlation between members' errors. An algorithm that generates either substandard members or members with correlated errors has room for significant improvement.

References

[1] A. Avizienis and J. B. J. Kelly. Fault tolerance by design diversity: Concepts and experiments. In *IEEE Computer*, volume 17(8), pages 67–80, 1984.

[2] Markus Brameier and Wolfgang Banzhaf. Evolving teams of predictors with linear genetic programming. *Genetic Programming and Evolvable Machines*, 2(4):381–408, 2001.

[3] L. Breiman. Bagging predictor, technical report 421. Technical report, University of California Berkley, 1994.

[4] Erick Cantu-Paz and Chandrika Kamath. Inducing oblique decision trees with evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7(1):54–68, 2003.

[5] R. Feldt. Generating multiple diverse software versions with genetic programming. In *Proceedings of the 24th EUROMICRO Conference, Workshop on Dpendable Computing Systems*, pages 387–396, 1998.

[6] L. Hatton. N-version vs. one good program. In *IEEE Software*, volume 14(6), pages 71–76, 1997.

[7] Thomas Haynes, Sandip Sen, Dale Schoenefeld, and Roger Wainwright. Evolving a team. In E. V. Siegel and J. R. Koza, editors, *Working Notes for the AAAI Symposium on Genetic Programming*, pages 23–30, MIT, Cambridge, MA, USA, 10–12 November 1995. AAAI.

[8] V. Hilford, M. R. Lyu, B. Cukie, A. Jamoussi, and F. B. Bastani. Diversity in the software development process. In *Proceedings of WORDS'97*, 1997.

[9] 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 L. Riolo, editors, *Genetic Programming 1997: Proceedings of the Second Annual Conference*, pages 195–200, Stanford University, CA, USA, 13-16 July 1997. Morgan Kaufmann.

[10] Hitoshi Iba. Bagging, boosting, and bloating in genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference: GECCO-1999*, pages 1053–1060. Morgan Kaufmann, 1999.

[11] Kosuke Imamura, Robert B. Heckendorn, Terence Soule, and James A. Foster. *N*-version genetic programming via fault masking. In James A. Foster, Evelyne Lutton, Julian Miller, Conor Ryan, and Andrea G. B. Tettamanzi, editors, *Genetic Programming, Proceedings of the 5th European Conference, EuroGP 2002*, volume 2278 of *LNCS*, pages 172–181, Kinsale, Ireland, 3-5 April 2002. Springer-Verlag.

[12] Kosuke Imamura, Terence Soule, Robert B. Heckendorn, and James A. Foster. Behavioral diversity and a probabilistically optimal GP ensemble. *Genetic Programming and Evolvable Machines*, 4(3):235–253, September 2003.

[13] J. C. Knight and N. B. Leveson. An experimental evaluation of the assumption of independence in multiversion programming, 1986.

[14] R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence (ICJA)*, pages 1137–1145. Morgan Kaufmann, 1995.

[15] John Koza. A genetic approach to the truck backer upper problem and the inter-twined spiral problem. In *Proceedings of IJCNN International Joint Conference on Neural Networks*, pages 310–318. IEEE Press, 1992.

[16] Yong Liu, Xin Yao, and Tetsuya Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.

[17] Sean Luke and Lee Spector. Evolving teamwork and coordination with genetic programming. In John R. Koza, David E. Goldberg, David B. Fogel, and Rick L. Riolo, editors, *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 150–156, Stanford University, CA, USA, 28–31 July 1996. MIT Press.

[18] R. Maclin and D. Optiz. An empirical evaluation of bagging and boosting. In *Proceedings of the 14th International Conference on Artificial Intelligence*, pages 546–551. AAAI Press/MIT Press, 1999.

[19] Imran Maqsood, Muhammad Raiz Khan, and Ajith Abraham. An ensemble of neural networks for weather prediction. *Neural Computing and Applications*, 13(2):112–123, 2004.

[20] D. W. Obitz, S. C. Basak, and B. D. Gute. Hazard assessment modeling: An evolutionary ensemble approach. In *Proceedings of the Genetic and Evolutionary Computation Conference: GECCO-1999*, pages 1543–1650. Morgan Kaufmann, 1999.

[21] K. Peng, S. Vucetic, P. Radivojac, C.J. Brown, A.K. Dunker, and Z. Obradovic. Optimizing long intrinsic disorder predictors with protein evolutionary information. *Journal of Bioinformatics and Computational Biology*, 3(1):1–26, 2004.

[22] Michael Defoin Platel, Malik Chami, Manuel Clergue, and Philippe Collard. Teams of genetic predictors for inverse problem solving. In *Proceeding of the 8th European Conference on Genetic Programming - EuroGP 2005*, 2005.

[23] 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.

[24] R. E. Schapire and Y. Freund. A short introduction to boosting. In *Journal of the Japanese Society for Artificial Intelligence*, volume 14(5), pages 771–780, 1999.

[25] 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.

[26] Terence Soule. Heterogeneity and specialization in evolving teams. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, pages 778–785, Las Vegas, Nevada, USA, 2000. Morgan Kaufmann.

[27] Terence Soule. Cooperative evolution on the intertwined spirals problem. In *Genetic Programming: Proceedings of the 6th European Conference on Genetic Programming, EuroGP 2003*, pages 434–442. Springer-Verlag, 2003.

[28] Gerhard Widmer. Discovering simple rules in complex data: a meta-learning algorithm and some surprising musical discoveries. *Artificial Intelligence*, 146:129–148, 2003.

[29] B. T. Zang and J. G. Joung. Enhancing robustness of genetic programming at the species level. In *Proceedings of the 2nd Annual Conference on Genetic Programming*, pages 336–342. Morgan Kaufmann, 1997.