# Novel Ways of Improving Cooperation and Performance in Ensemble Classifiers*

Russell Thomason
University of Idaho
PO Box 441010
Moscow, ID 83844-1010
thom0398@uidaho.edu

Terence Soule
University of Idaho
PO Box 441010
Moscow, ID 83844-1010
tsoule@cs.uidaho.edu

## ABSTRACT

There are two common methods of evolving teams of genetic programs. Research suggests Island approaches produce teams of strong individuals that cooperate poorly and Team approaches produce teams of weak individuals that cooperate strongly. Ideally, teams should be composed of strong individuals that cooperate well. In this paper we present a new class of algorithms called Orthogonal Evolution of Teams (OET) that overcomes the weaknesses of current Island and Team approaches by applying evolutionary pressure at both the level of teams and individuals during selection and replacement. We present four novel algorithms in this new class and compare their performance to Island and Team approaches as well as multi-class Adaboost on a number of classification problems.

## Categories and Subject Descriptors

I.2.2 [**Artificial Intelligence**]: Automatic Programming

## General Terms

Algorithms, Performance

## Keywords

Genetic Programming, Performance Analysis

## 1. INTRODUCTION

Many real world problems are too complex to expect a single genetic program to solve adequately. A single genetic program will tend to find the general solution and overlook specialized subdomains within the larger problem space and thus to make errors on those subdomains. Therefore considerable research has gone into evolving teams, or ensembles, of genetic programs that use a cooperation mechanism, such as voting, to produce an answer. This allows members to specialize on distinct subdomains and reduce overall errors.

*This research supported by NSF Grant 0535130.


*GECCO'07,* July 7–11, 2007, London, England, United Kingdom.


The extent of cooperation is how correlated the errors, or faults, of the team members are. If members have correlated faults then they will answer the same test cases incorrectly and little is gained by forming a team. If members have independent faults, by evolving teams using the Island approach for example, then forming a team increases performance because subgroups of the team can mask the faults of other members. Ideally, members have inversely correlated faults which means that members optimize their ability to mask the faults of others by specializing in different subdomains of the problem space. This can only occur if team members are specifically designed, or evolved, to have this property. Any approach that produces the members independently, e.g. Island approaches or N-version programming [9, 6], can at best be expected to produce independent faults.

Soule introduced a novel class of algorithms referred to as Orthogonal Evolution of Teams (OET) that are designed to apply direct pressure to both teams and their members [10]. Soule described this class of algorithms as orthogonal because they alternate between two orthogonal views of the population: as a single population of teams of size $N$ and as a set of $N$ independent populations of individuals. Research on OET algorithms made it apparent that Island and Team approaches only consider some of the possible ways of thinking about selection and replacement. Island approaches apply evolutionary pressure solely at the level of individuals and Team approaches apply evolutionary pressure solely at the level of teams. OET algorithms encompass additional ways of applying evolutionary pressure during selection and replacement that works on both levels.

In this paper we present a more thorough comparison of algorithms to evolve teams, including Island and Team approaches, four variations of OET, and Adaboost. We used six classification problems, five from the UCI machine learning repository [3] and the intertwined spirals problem, although space only allows us to present detailed results from two problems. Our results show that OET algorithms do combine the strengths of Team and Island approaches. We show that OET algorithms increase both member performance and team cooperation which leads to higher overall team performance. We show that OET gives precise control over the amount of pressure to apply to teams and members and finding the right balance is important for each problem. Finally, our results show that balancing team size and population size is fundamental to optimizing performance.

## 2. BACKGROUND

Island approaches evolve independent populations to cre-

ate members with independent faults. Traditionally, Island approaches are used when individuals can perform reasonably well alone, e.g. classification problems where a single classifier can perform well although a team of classifiers may perform better. N-Version Genetic Programming (NVGP) developed by Imamura et al. evolves $N$ independent populations (islands) and when evolution halts one random individual is drawn from each population to create a team of $N$ individuals [8]. Although NVGP produces some teams with independent errors for a range of problems, the majority of teams do not have independent faults. In general, it takes a large number of random draws to form a team with independent faults. This appears to invalidate the assumption often made for Island approaches that independent evolutionary runs will consistently generate solutions with independent errors.

In Team approaches a single population of teams evolves, where each team is a collection of $N$ members. Typically, Team approaches are used when the problem requires a team, e.g. Serengeti world [11, 7] where $N$ lions must work together to capture a gazelle or in the robocup competition. Studies of the Team approach strongly suggest that it avoids correlated errors; each member evolves to have few errors on a particular domain of the problem and the domain for each member is different [12, 2]. Thus, the whole team has very few unmasked errors. This is a reasonable result as all of the selective pressure is on the team so there is direct pressure to evolve members that mask each others' errors. Unfortunately, in Team based methods 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 [13, 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.

In [10] Soule introduced a new class of algorithms called Orthogonal Evolution of Teams (OET). OET is a class of cooperative, co-evolutionary algorithms that evolves ensemble members with distinct areas of specialization. Only one algorithm from this class was studied; it was compared to Island and Team approaches where the test problems were simple, yet illustrative. In that work performance was not compared to boosting techniques.

Using the expected failure rate model [1] it was shown that Team approaches create teams with very high levels of cooperation, where members have inversely correlated faults [10]. Unfortunately the members have relatively poor fitnesses [14] so the results were not better than other methods. In contrast it was confirmed that Island approaches tend to create teams with relatively fit members, but they do not cooperate as well [8]. It was also shown that Orthogonal Evolution of Teams (OET) overcomes these weaknesses [10]. The OET algorithm combined the advantages of both the Island and Team approaches, generating highly successful team members that cooperated well within their team, leading to robust solutions that cover an entire problem domain with few gaps or errors, if any.

One of the most successful attempts at developing ensemble based classifiers is the Adaboost algorithm, developed by Freund and Schapire [5, 4]. It builds an ensemble by training each member to focus on the test cases that have not been classified correctly yet. A weight is assigned to each training case and they are initially all equal. After the first classifier in the ensemble is trained, the weights are re-adjusted so the cases missed have a higher weight for the next classifier. This process continues until the ensemble is complete.

Adaboost was shown to guarantee an increase in fitness with an increase in team size, assuming the training algorithm produced individual classifiers which could classify correctly at least 50% of the time [5]. For a two class problem, this means the training algorithm only has to do slightly better than random guessing. However, for a multi-class problem, this level of accuracy becomes more unrealistic as the number of classes increases. To address this problem, Zhu et al. developed multi-class Adaboost [15]. The algorithm is a modified version of the Adaboost which only requires classifiers to perform better than random guessing, no matter the number of classes.

However, one limitation of boosting techniques, including Adaboost, is that they require that the training process be divided into distinct cases that can be assigned specific weights and that members can be trained individually (as in the Island approach). Thus, boosting cannot be used to evolve teams for Serengeti world or robocup. These types of problems, where there are not distinct training cases and the problem is only solvable by a team, have been the traditional domain of the Team approaches described above.

# 3. ALGORITHMS

There are four combinations for doing selection and replacement when applying pressure at the level of teams and individuals. First there is team selection with team replacement (TT) which defines current Team approaches. Second there is individual selection with individual replacement (II) which defines current Island approaches. In addition, it is possible to mix them; individual selection with team replacement (IT) which we define as OET1, and team selection with individual replacement (TI) which we define as OET2, both of which are described in detail below.

In OET1 selection is done on individuals and replacement is done on teams. Offspring are created by making an empty team and adding highly fit individuals one at a time by treating the single population of $N$-sized teams as $N$ independent populations of individuals and doing tournament selection within each population. Therefore, the first team member is chosen from the population that represents all of the first members from each team; the second team member is chosen from the population that represents all of the second members from each team and this continues until the new team is filled. Two teams are constructed in this way and they undergo crossover and mutation. The offspring are evaluated as teams and replacement is done by comparing team fitness, so poor teams are selected for replacement. A member must have high fitness to be selected for a parent team and a team in the population must have a high fitness to avoid being selected for replacement.

Conversely, in OET2 selection is done on teams and replacement is done on individuals. Two highly fit teams are selected to be parents by tournament selection and they undergo crossover and mutation to produce two new offspring teams. Replacement is done by comparing the fitness of individual team members in the offspring to the fitness of individual team members in the population. This is done by treating the population of $N$-sized teams as $N$ independent populations of individuals where poor individuals are selected for replacement by individuals in the new offspring.

A team must have high fitness to be selected as a parent, meaning its members are cooperating well, and a team member must have a high fitness to avoid being selected for replacement.

Both of these approaches (OET1 and OET2) create evolutionary pressure for team members to perform well and for their teams to perform well. However, these four *basic* algorithms do not have to be used in isolation during the entire evolutionary run. There are many possible *combination* algorithms that can be created by alternating the use of the four *basic* algorithms, for example, using one algorithm during even iterations and using another algorithm during odd iterations. We present two such examples, one that alternates between Island (II) and Team (TT) and a second that alternates between OET1 (IT) and Team (TT). In the former there is equal pressure on individuals and teams (II/TT) and in the later there is more pressure on teams (IT/TT). Algorithms created in this way can more precisely tune the amount of pressure that can be applied to improving individual performance versus improving team cooperation. We found that the optimal amount of pressure for teams and individuals is problem dependent so this freedom is important to optimizing performance.

# 4. METHODS

## 4.1 Parameters

A simple steady state population model is used in our genetic program. The parameters are summarized in Table 1. We ran three sets of experiments with 90000, 180000, and 270000 evaluations each. An evaluation is defined as the evaluation of a single function tree, not an entire team. Thus, evaluating a team of $N$ members uses $N$ evaluations.

We tested team sizes of 2, 3, 4, and 5. For each set of parameters 30 trials were performed and each trial ran for 150 iterations. In order to keep the number of evaluations fixed, the population size had to vary. It is equal to the number of evaluations divided by the product of team size and iterations. All of the algorithms undergo crossover and mutation $populationsize/2$ times each iteration because two offspring are made each iteration.

## 4.2 Teams and Voting

A team is a collection of function trees. Each member in a team produces a vote by generating a single real valued output that is mapped to a particular classification. The vote that a member makes is weighted, where the weight of the vote is equal to the member's fitness (between 0 and 1). The final classification of the ensemble is the classification with the most weight.

A portion of the real number line is partitioned into equal-sized 'buckets', one for each possible classification. If the value a member produces falls inside one of the buckets, that becomes its vote. If the value falls outside any of the buckets, its vote is treated as a classification of "I don't know" (IDK). Therefore, the fitness function of a member depends on the number of correct classifications, incorrect classifications, and IDK classifications. The member fitness function is:

$$Fitness_{\text{mem}} = \left( \frac{V_{\text{correct}}}{V_{\text{correct}} + V_{\text{wrong}} + V_{\text{idk}} * P_{\text{idk}}} \right) \quad (1)$$

$V_{\text{correct}}$ is the number of votes that are correct classifications. $V_{\text{wrong}}$ is the number of votes that are incorrect classifications. $V_{\text{idk}}$ is the number of IDK classifications, and $P_{\text{idk}}$ is a real value in the interval [0, 1]. If this value is 0, then voting IDK is not penalized because the IDK term becomes 0. If the value is 1, then voting IDK is the same as voting incorrectly, and if it is anywhere in between, casting an IDK vote still reduces fitness, but not as much as voting incorrectly. A value less than 1 encourages members to not vote if their vote will hurt the team fitness. A value greater than 0 discourages members from picking a small number of test cases, voting perfectly on them and never trying to classify anything else (by voting IDK many times). However, the team always needs to make a final classification so in the rare case that all of the team members vote IDK, the majority class is returned as the team's classification. Otherwise, the classification the team makes is the classification that received the largest weighted sum of votes. The team fitness function is:

$$Fitness_{\text{team}} = \left( \frac{C_{\text{correct}}}{C_{\text{correct}} + C_{\text{wrong}}} \right) \quad (2)$$

$C_{\text{correct}}$ is the number of classifications that are correct and $C_{\text{wrong}}$ is the number of incorrect classifications.

In addition to recording team and member fitnesses using the fitness functions just described, we also recorded the percent of the classifications that each team member answered correctly. This data is what is used in the tables that compare the average member performance in all of the algorithms.

## 4.3 Experiments

We ran our experiments on a total of six classification problems, five from the UCI machine learning repository (the data sets are E.coli, Heart Disease, Yeast, Wine, and Iris) and also intertwined spirals. Due to space limitations we are only including data from E.coli and Heart Disease. We chose classification problems because they are easy to analyze and allow direct comparison to Island and boosting techniques. It also allows us to measure member performance in order to compare how this varies among the different algorithms.

### *4.3.1 E.coli Data Set*

This classification problem uses the E.coli data set from the UCI machine learning repository [3]. The goal is to predict the cellular localization sites of proteins found in E.coli. There are 336 classification instances, each composed of 7 predictive attributes and a name. There are 8 different classifications which represent the localization sites of the proteins. Each class is represented unevenly, the largest group comprising 143 of the instances and the smallest two groups with 2 instances each.

### *4.3.2 Heart Disease Data Set - Cleveland*

This classification problem uses the Cleveland Heart Disease data set found on the UCI machine learning repository [3]. The goal is to predict the presence of heart disease using 13 predictive attributes. There are five classes of heart disease and the scale ranges from 0, which represents no heart disease, to 4, which represents severe heart disease. Previous studies have only made a binary distinction for the presence of heart disease while we used all 5 classes to distinguish the performance of the algorithms.

| **Problems** | E.coli | Heart Disease |
|---|---|---|
| **Training Cases** | random 50% | |
| **Testing Cases** | random 50% | |
| **Member Fitness** | $V_{\text{correct}}$ / ($V_{\text{correct}}$ + $V_{\text{wrong}}$ + $V_{\text{idk}}$ * $P_{\text{idk}}$) | |
| **Team Fitness** | $C_{\text{correct}}$ / ($C_{\text{correct}}$ + $C_{\text{wrong}}$) | |
| **Function Set** | +, -, *, /, IFLTE | |
| **Terminal Set** | 7 attributes and CONST | 13 attributes and CONST |
| **Evaluations** | 90000, 180000, 270000 | |
| **Team Sizes** | 2, 3, 4, 5 | |
| **Iterations** | 150 | |
| **Population Size** | evaluations / (team size * iterations) | |
| **Bucket Size** | 2.0 | |
| **Mutation Amount** | abs of random normal (mean 2.0 and stdev 2.0) | |
| **Selection** | 3 member tournament | |
| **Initial Population** | ramped half and half | |
| **Number of trials** | 30 | |

**Table 1: Summary of the evolutionary algorithm parameters.**

# 5. RESULTS

## 5.1 E.coli

Table 2 shows average team training fitness and Table 3 shows average team testing fitness. In general, Island performed slightly worse than Team and both were outperformed by all of the OET algorithms. OET1 and OET1/Team did particularly well. Although some performance differences are within the standard deviations, the trend across all parameter combinations is significant and shows that the OET algorithms have the best performance. In both tables the best performance for each algorithm is emphasized.

Unsurprisingly, the best results were achieved with the maximum number of evaluations. However, there appears to be a tradeoff between population size and team size. When looking at the results for the Island algorithm, ensembles of size 2 did the best with 90000 evaluations, ensembles of size 3 did the best with 180000 evaluations and ensembles of size 4 did the best with 270000 evaluations. This suggests that there is a critical population size necessary to achieve optimum results and once that level is reached additional evaluations can be used to make the team size larger. The OET algorithms also showed that larger ensembles perform better as long as the population size is large enough to support them. Team ensembles performed the best with ensembles of size 2 in all cases showing that it did better with the largest population size possible.

### *5.1.1 Member Fitness versus Cooperation*

The most interesting results come from comparing average member training fitness in Table 4 and average member testing fitness in Table 5 across all the algorithms. As expected the individuals comprising Island ensembles had relatively high fitness while the individuals comprising Team ensembles had relatively low fitness. Since the overall ensemble performances were very similar, this confirms that individuals in Team ensembles must be cooperating significantly better than individuals in Island ensembles.

All of the OET algorithms had average member fitnesses that are better than those in Team ensembles. Since OET1, Island/Team and OET1/Team each had average member fitnesses that were lower than Island, but their teams outperformed Island, this shows that the OET algorithms lead to more cooperation than in the Island algorithm.

Interestingly, OET2 produced average member fitness better than those in Island ensembles. We were surprised by this last finding because in Island ensembles all of the evolutionary pressure is placed on individuals. This suggests that selection for replacement is more important than selection to be a parent because in OET2 the pressure on individuals comes during the replacement phase. Possibly this is because most offspring will be less fit than their parents are due to the destructive effects of crossover.

### *5.1.2 Boosting Results*

The average team training and average team testing performance for Adaboost is shown in Table 6. Note that the team sizes are quite small for boosting (in order to make direct comparisons). For teams of size 4 and 5 the training fitness is clearly better than any of the other algorithms. However, the testing performance of Adaboost is equivalent to the OET algorithms. Not surprisingly Adaboost does better with the larger teams, but the OET algorithms do better with smaller teams. Thus if team size is a consideration OET may be preferable. Adaboost also shows a larger difference between training and testing fitness which suggests over fitting. This could be a problem with noisy data sets.

## 5.2 Heart Disease

Table 7 shows average team training fitness and Table 8 shows average team testing fitness. This data set shows the same trends as the E.coli problem. The Island and Team algorithms perform very similarly while the OET algorithms outperform them both. In this case OET2 does particularly well.

Average individual training fitness can be seen in Table 9 and average individual testing fitness can be seen in Table 10. Again the results are similar to the previous problem. The Island algorithm produces relatively high fit individuals while the Team algorithm shows relatively poor individuals. All the OET algorithms produce member more fit than members in the Team algorithm and OET2 produces members more fit than those found in Island ensembles.

## 5.3 Other Problems

Space limitations prevented us from showing data from the other problems (Yeast, Iris, Wine, and Intertwined Spirals). It should be noted that some classification problems such as Iris and Wine are simply too easy to show a large performance difference between the algorithms. However, on problems that are more difficult the general trends are similar to the E.coli and Heart disease problems.

# 6. CONCLUSION

This research has shown that current Island and Team approaches to evolving ensembles are lacking. Team algorithms do not directly apply pressure on team members to increase their fitness, which leads to relatively poor member performance. Island approaches do not directly apply pressure on team members to cooperate, leading to suboptimal team performance. OET algorithms apply direct evolutionary pressure to both members and teams leading to increased overall performance. OET algorithms produce ensembles with member fitness near that of Island approaches that cooperate at levels near Team approaches. We also provided several example algorithms from this class and a framework for creating additional algorithms by combining the four *basic* algorithms (Island, Team, OET1, OET2).

Our results show that the ideal balance between team and individual pressure depends on the problem since OET1/Team performed the best on the E.coli problem and OET2 performed the best on the Heart Disease problem. Thus, one significant advantage of the OET algorithms is that it is possible to more closely control the desired fitness of individuals and the level of cooperation within ensembles. Clearly, applying pressure at both levels is advantageous, but if in a given problem cooperation is more important then it is possible to use algorithms that apply more team pressure, such as alternating between OET1 and Team. Conversely, if a problem favors individual performance, e.g. there are fewer distinct subdomains to identify, then it is possible to choose an algorithm that places more pressure on member performance.

Adaboost does not allow this same level of control. However, Adaboost did produce the best training results and was on par with OET on the testing results. The results also show that boosting techniques work better with bigger teams. The most significant drawback to boosting techniques (which was not fully explored here) is that they require the problem be separable into independent training cases. This applies to classification problems, but not, for example, controlling robot teams. The OET1 and OET2 algorithms can be applied to non-separable problems as long as a fitness can be given to team members, even if the team must be evaluated as a whole. Future work will involve comparing Team, OET1, and OET2 algorithms on team oriented problems.

It was also shown that, for a fixed number of evaluations, using larger ensembles is not always optimal, particularly if it causes the population size to become too small. This was clearly the case for the Island algorithm on the E.coli problem - with fewer total evaluations smaller teams and larger populations performed better than larger teams and smaller populations. Thus, in using any approach to evolve teams it is important to optimize the balance between population and team size.

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

[3] C. B. D.J. Newman, S. Hettich and C. Merz. UCI repository of machine learning databases, 1998.

[4] S. R. E. Freund, Yoav. Experiments with a new boosting algorithm. In *Machine Learning: Proceedings of the Thirteenth International Conference*, 1996.

[5] S. R. E. Freund, Yoav. A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence*, 14:771–780, 1999.

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

[7] S. S. D. S. Haynes, T. and R. Wainwright. Evolving a team. In Working Notes of the AAAI-95 Fall Symposium on Genetic Programming, 1995.

[8] K. Imamura, R. B. Heckendorn, T. Soule, and J. A. Foster. Behavioral diversity and a probabilistically optimal gp ensemble. *Genetic Programming and Evolvable Machines*, 4:235–253, 2004.

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

[10] P. Komireddy and T. Soule. *Orthogonal Evolution of Teams: A Class of Algorithms for Evolving Teams with Inversely Correlated Errors*. 2006.

[11] S. Luke and L. Spector. Evolving teamwork and coordination with genetic programming. In *In Genetic Programming 1996: Proceedings of the First Annual Conference..*, pages 141–149. Cambridge: MIT Press, 1996.

[12] T. Soule. Voting teams: A cooperative approach to non-typical problems. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela, and R. E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 916–922, Orlando, Florida, USA, 13-17 July 1999. Morgan Kaufmann.

[13] T. 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.

[14] T. 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.

[15] R. S. H. T. Zhu, Ji. Multi-class adaboost. *Technical Report 430, Department of Statistics, University of Michigan*, 2005.

| Evals | Team Size | Pop Size | Island | Team | OET1 | OET2 | Island/ Team | OET1/ Team |
|---|---|---|---|---|---|---|---|---|
| 90000 | 2 | 300 | 0.724 (0.079) | 0.771 (0.063) | 0.784 (0.076) | 0.778 (0.064) | 0.797 (0.065) | 0.819 (0.068) |
| 90000 | 3 | 200 | 0.715 (0.075) | 0.757 (0.077) | 0.788 (0.051) | 0.771 (0.076) | 0.785 (0.066) | 0.788 (0.081) |
| 90000 | 4 | 150 | 0.708 (0.063) | 0.728 (0.066) | 0.765 (0.065) | 0.768 (0.065) | 0.776 (0.069) | 0.736 (0.066) |
| 90000 | 5 | 120 | 0.760 (0.072) | 0.706 (0.070) | 0.746 (0.066) | 0.768 (0.060) | 0.773 (0.070) | 0.734 (0.070) |
| 180000 | 2 | 600 | 0.720 (0.081) | 0.813 (0.067) | 0.829 (0.058) | 0.800 (0.065) | 0.808 (0.054) | 0.835 (0.045) |
| 180000 | 3 | 400 | 0.768 (0.082) | 0.781 (0.070) | 0.827 (0.053) | 0.836 (0.047) | 0.810 (0.060) | 0.840 (0.058) |
| 180000 | 4 | 300 | 0.777 (0.075) | 0.782 (0.074) | 0.815 (0.049) | 0.827 (0.073) | 0.805 (0.055) | 0.820 (0.056) |
| 180000 | 5 | 240 | 0.762 (0.077) | 0.730 (0.066) | 0.788 (0.065) | 0.829 (0.053) | 0.812 (0.049) | 0.783 (0.081) |
| 270000 | 2 | 900 | 0.745 (0.093) | **0.826** (0.063) | 0.842 (0.048) | 0.843 (0.048) | 0.814 (0.064) | 0.850 (0.049) |
| 270000 | 3 | 600 | 0.758 (0.088) | 0.821 (0.064) | **0.859** (0.036) | 0.832 (0.056) | 0.813 (0.070) | 0.858 (0.062) |
| 270000 | 4 | 450 | **0.807** (0.068) | 0.805 (0.070) | 0.846 (0.041) | **0.848** (0.048) | **0.838** (0.064) | **0.861** (0.060) |
| 270000 | 5 | 360 | 0.798 (0.059) | 0.776 (0.076) | 0.831 (0.062) | 0.837 (0.057) | 0.819 (0.066) | 0.829 (0.072) |

**Table 2: Average team training fitness on the E.coli problem (150 iterations).**

| Evals | Team Size | Pop Size | Island | Team | OET1 | OET2 | Island/ Team | OET1/ Team |
|---|---|---|---|---|---|---|---|---|
| 90000 | 2 | 300 | 0.639 (0.060) | 0.694 (0.077) | 0.701 (0.061) | 0.690 (0.054) | 0.697 (0.063) | 0.721 (0.064) |
| 90000 | 3 | 200 | 0.639 (0.051) | 0.685 (0.070) | 0.696 (0.062) | 0.697 (0.062) | 0.699 (0.056) | 0.693 (0.051) |
| 90000 | 4 | 150 | 0.646 (0.037) | 0.669 (0.070) | 0.696 (0.055) | 0.692 (0.063) | 0.697 (0.051) | 0.681 (0.056) |
| 90000 | 5 | 120 | 0.689 (0.071) | 0.665 (0.056) | 0.683 (0.059) | 0.684 (0.059) | 0.703 (0.063) | 0.661 (0.079) |
| 180000 | 2 | 600 | 0.630 (0.054) | 0.722 (0.057) | 0.729 (0.057) | 0.690 (0.060) | 0.703 (0.058) | 0.726 (0.061) |
| 180000 | 3 | 400 | 0.672 (0.061) | 0.680 (0.075) | 0.724 (0.059) | 0.719 (0.065) | 0.704 (0.064) | 0.707 (0.076) |
| 180000 | 4 | 300 | 0.701 (0.067) | 0.676 (0.081) | 0.723 (0.065) | 0.723 (0.068) | 0.714 (0.058) | 0.717 (0.072) |
| 180000 | 5 | 240 | 0.682 (0.069) | 0.676 (0.065) | 0.715 (0.068) | 0.722 (0.064) | 0.718 (0.054) | 0.706 (0.066) |
| 270000 | 2 | 900 | 0.648 (0.059) | 0.713 (0.073) | 0.734 (0.050) | 0.718 (0.063) | 0.712 (0.051) | 0.730 (0.064) |
| 270000 | 3 | 600 | 0.659 (0.065) | **0.725** (0.072) | 0.744 (0.046) | 0.726 (0.061) | 0.707 (0.072) | **0.744** (0.046) |
| 270000 | 4 | 450 | **0.704** (0.051) | 0.722 (0.066) | **0.744** (0.056) | 0.730 (0.056) | **0.724** (0.057) | 0.733 (0.066) |
| 270000 | 5 | 360 | 0.699 (0.063) | 0.686 (0.076) | 0.729 (0.054) | **0.735** (0.056) | 0.704 (0.065) | 0.729 (0.071) |

**Table 3: Average team testing fitness on the E.coli problem (150 iterations).**

| Evals | Team Size | Pop Size | Island | Team | OET1 | OET2 | Island/Team | OET1/Team |
|---|---|---|---|---|---|---|---|---|
| 90000 | 2 | 300 | 0.6598 | 0.4641 | 0.5555 | 0.6889 | 0.6650 | 0.5460 |
| 90000 | 3 | 200 | 0.6359 | 0.3876 | 0.5531 | 0.6561 | 0.6090 | 0.5325 |
| 90000 | 4 | 150 | 0.6213 | 0.2659 | 0.5229 | 0.6355 | 0.6078 | 0.4030 |
| 90000 | 5 | 120 | 0.6415 | 0.2253 | 0.4716 | 0.6282 | 0.5781 | 0.3794 |
| 180000 | 2 | 600 | 0.6616 | 0.5020 | 0.6258 | 0.7050 | 0.6778 | 0.6045 |
| 180000 | 3 | 400 | 0.6645 | 0.3248 | 0.5828 | 0.6863 | 0.5959 | 0.5303 |
| 180000 | 4 | 300 | 0.6546 | 0.2873 | 0.5483 | 0.6711 | 0.5980 | 0.5321 |
| 180000 | 5 | 240 | 0.6421 | 0.2332 | 0.5305 | 0.6534 | 0.5947 | 0.4468 |
| 270000 | 2 | 900 | 0.6857 | 0.4894 | 0.6490 | 0.7393 | 0.6813 | 0.6013 |
| 270000 | 3 | 600 | 0.6590 | 0.3732 | 0.6297 | 0.6932 | 0.6424 | 0.5915 |
| 270000 | 4 | 450 | 0.6659 | 0.3032 | 0.5853 | 0.6751 | 0.6260 | 0.5253 |
| 270000 | 5 | 360 | 0.6530 | 0.2356 | 0.5368 | 0.6672 | 0.6233 | 0.4732 |

**Table 4: Average member training fitness on the E.coli problem (150 iterations).**

| Evals | Team Size | Pop Size | Island | Team | OET1 | OET2 | Island/Team | OET1/Team |
|---|---|---|---|---|---|---|---|---|
| 90000 | 2 | 300 | 0.5815 | 0.4285 | 0.5129 | 0.6140 | 0.5967 | 0.5009 |
| 90000 | 3 | 200 | 0.5646 | 0.3653 | 0.5091 | 0.5956 | 0.5551 | 0.5009 |
| 90000 | 4 | 150 | 0.5614 | 0.2516 | 0.4897 | 0.5807 | 0.5516 | 0.3850 |
| 90000 | 5 | 120 | 0.5754 | 0.2172 | 0.4530 | 0.5708 | 0.5313 | 0.3623 |
| 180000 | 2 | 600 | 0.5803 | 0.4547 | 0.5610 | 0.6198 | 0.6015 | 0.5531 |
| 180000 | 3 | 400 | 0.5791 | 0.3044 | 0.5382 | 0.6079 | 0.5384 | 0.4858 |
| 180000 | 4 | 300 | 0.5814 | 0.2696 | 0.5115 | 0.5998 | 0.5388 | 0.4975 |
| 180000 | 5 | 240 | 0.5733 | 0.2262 | 0.4988 | 0.5846 | 0.5420 | 0.4176 |
| 270000 | 2 | 900 | 0.5922 | 0.4406 | 0.5851 | 0.6421 | 0.6026 | 0.5369 |
| 270000 | 3 | 600 | 0.5712 | 0.3452 | 0.5734 | 0.6160 | 0.5728 | 0.5506 |
| 270000 | 4 | 450 | 0.5871 | 0.2839 | 0.5393 | 0.6012 | 0.5586 | 0.4859 |
| 270000 | 5 | 360 | 0.5779 | 0.2213 | 0.5008 | 0.5942 | 0.5550 | 0.4424 |

**Table 5: Average member testing fitness on the E.coli problem (150 iterations).**

| Evals | Iterations | Team Size | Pop Size | Avg Training | Avg Testing |
|---|---|---|---|---|---|
| 90000 | 150 | 2 | 300 | 0.7024 | 0.5902 |
| 90000 | 150 | 3 | 200 | 0.8013 | 0.6708 |
| 90000 | 150 | 4 | 150 | 0.8611 | 0.6949 |
| 90000 | 150 | 5 | 120 | 0.8844 | 0.7259 |
| 180000 | 150 | 2 | 600 | 0.7431 | 0.6276 |
| 180000 | 150 | 3 | 400 | 0.8419 | 0.6996 |
| 180000 | 150 | 4 | 300 | 0.8903 | 0.7108 |
| 180000 | 150 | 5 | 240 | 0.9058 | 0.7393 |
| 270000 | 150 | 2 | 900 | 0.7229 | 0.5845 |
| 270000 | 150 | 3 | 600 | 0.8725 | 0.6998 |
| 270000 | 150 | 4 | 450 | 0.9010 | 0.7295 |
| 270000 | 150 | 5 | 360 | **0.9228** | **0.7476** |

**Table 6: Adaboost results on the E.coli problem (also averaged over 30 trials).**

| Evals | Team Size | Pop Size | Island | Team | OET1 | OET2 | Island/ Team | OET1/ Team |
|---|---|---|---|---|---|---|---|---|
| 90000 | 2 | 300 | 0.656 (0.044) | 0.700 (0.030) | 0.722 (0.033) | 0.724 (0.034) | 0.718 (0.032) | 0.725 (0.035) |
| 90000 | 3 | 200 | 0.670 (0.039) | 0.696 (0.022) | 0.728 (0.025) | 0.736 (0.027) | 0.707 (0.029) | 0.728 (0.033) |
| 90000 | 4 | 150 | 0.690 (0.040) | 0.707 (0.026) | 0.719 (0.030) | 0.756 (0.030) | 0.727 (0.029) | 0.729 (0.030) |
| 90000 | 5 | 120 | 0.693 (0.040) | 0.698 (0.027) | 0.723 (0.031) | 0.754 (0.029) | 0.717 (0.023) | 0.724 (0.026) |
| 180000 | 2 | 600 | 0.667 (0.038) | 0.726 (0.031) | 0.738 (0.033) | 0.763 (0.035) | 0.725 (0.043) | 0.741 (0.036) |
| 180000 | 3 | 400 | 0.687 (0.035) | 0.724 (0.038) | 0.744 (0.027) | 0.771 (0.033) | 0.744 (0.028) | 0.746 (0.030) |
| 180000 | 4 | 300 | 0.704 (0.046) | 0.723 (0.030) | 0.748 (0.030) | 0.770 (0.034) | 0.753 (0.022) | 0.750 (0.030) |
| 180000 | 5 | 240 | 0.729 (0.037) | 0.717 (0.026) | 0.741 (0.030) | 0.785 (0.026) | 0.748 (0.024) | 0.742 (0.024) |
| 270000 | 2 | 900 | 0.666 (0.039) | 0.733 (0.036) | 0.751 (0.028) | 0.770 (0.039) | 0.749 (0.028) | 0.755 (0.030) |
| 270000 | 3 | 600 | 0.714 (0.041) | 0.730 (0.037) | 0.751 (0.033) | 0.789 (0.027) | 0.754 (0.031) | 0.756 (0.034) |
| 270000 | 4 | 450 | 0.734 (0.036) | **0.734** (0.029) | 0.751 (0.028) | 0.782 (0.027) | 0.765 (0.029) | **0.765** (0.035) |
| 270000 | 5 | 360 | **0.738** (0.034) | 0.731 (0.027) | **0.760** (0.033) | **0.796** (0.033) | **0.767** (0.026) | 0.764 (0.026) |

**Table 7: Average team training fitness on the Heart Disease problem (150 iterations).**

| Evals | Team Size | Pop Size | Island | Team | OET1 | OET2 | Island/ Team | OET1/ Team |
|---|---|---|---|---|---|---|---|---|
| 90000 | 2 | 300 | 0.522 (0.040) | 0.551 (0.027) | 0.555 (0.027) | 0.550 (0.027) | 0.552 (0.031) | 0.555 (0.029) |
| 90000 | 3 | 200 | 0.530 (0.035) | 0.554 (0.029) | 0.555 (0.026) | 0.557 (0.028) | 0.557 (0.029) | 0.562 (0.031) |
| 90000 | 4 | 150 | 0.537 (0.031) | 0.555 (0.026) | 0.565 (0.025) | 0.559 (0.028) | 0.558 (0.031) | 0.556 (0.030) |
| 90000 | 5 | 120 | 0.541 (0.030) | 0.560 (0.036) | **0.568** (0.030) | 0.562 (0.026) | 0.555 (0.027) | 0.565 (0.026) |
| 180000 | 2 | 600 | 0.507 (0.034) | 0.551 (0.031) | 0.552 (0.039) | 0.551 (0.026) | 0.548 (0.037) | 0.554 (0.028) |
| 180000 | 3 | 400 | 0.529 (0.033) | 0.548 (0.034) | 0.562 (0.032) | 0.563 (0.031) | **0.558** (0.034) | 0.562 (0.032) |
| 180000 | 4 | 300 | 0.532 (0.034) | 0.555 (0.029) | 0.563 (0.026) | 0.557 (0.032) | 0.556 (0.035) | 0.566 (0.027) |
| 180000 | 5 | 240 | **0.547** (0.030) | 0.556 (0.030) | 0.566 (0.025) | 0.565 (0.029) | 0.556 (0.036) | 0.558 (0.024) |
| 270000 | 2 | 900 | 0.510 (0.043) | 0.557 (0.031) | 0.558 (0.035) | 0.552 (0.031) | 0.553 (0.034) | 0.553 (0.028) |
| 270000 | 3 | 600 | 0.528 (0.033) | 0.560 (0.024) | 0.559 (0.025) | 0.562 (0.030) | 0.555 (0.035) | 0.566 (0.028) |
| 270000 | 4 | 450 | 0.532 (0.035) | 0.555 (0.036) | 0.564 (0.031) | 0.557 (0.026) | 0.554 (0.034) | **0.566** (0.029) |
| 270000 | 5 | 360 | 0.542 (0.033) | **0.566** (0.025) | 0.564 (0.029) | **0.568** (0.023) | 0.557 (0.030) | 0.563 (0.029) |

**Table 8: Average team testing fitness on the Heart Disease problem (150 iterations).**

| Evals | Team Size | Pop Size | Island | Team | OET1 | OET2 | Island/Team | OET1/Team |
|---|---|---|---|---|---|---|---|---|
| 90000 | 2 | 300 | 0.5874 | 0.4847 | 0.5815 | 0.6257 | 0.6201 | 0.5782 |
| 90000 | 3 | 200 | 0.5707 | 0.4628 | 0.5624 | 0.5970 | 0.5796 | 0.5648 |
| 90000 | 4 | 150 | 0.5640 | 0.3676 | 0.5438 | 0.5880 | 0.5579 | 0.5301 |
| 90000 | 5 | 120 | 0.5625 | 0.3161 | 0.5305 | 0.5780 | 0.5380 | 0.5045 |
| 180000 | 2 | 600 | 0.5926 | 0.5171 | 0.6112 | 0.6557 | 0.6137 | 0.5980 |
| 180000 | 3 | 400 | 0.5831 | 0.4461 | 0.5765 | 0.6213 | 0.5843 | 0.5679 |
| 180000 | 4 | 300 | 0.5765 | 0.4070 | 0.5585 | 0.5937 | 0.5763 | 0.5476 |
| 180000 | 5 | 240 | 0.5738 | 0.3396 | 0.5497 | 0.5827 | 0.5631 | 0.5188 |
| 270000 | 2 | 900 | 0.5970 | 0.4570 | 0.6349 | 0.6627 | 0.6155 | 0.6094 |
| 270000 | 3 | 600 | 0.5989 | 0.4650 | 0.5858 | 0.6347 | 0.5972 | 0.5840 |
| 270000 | 4 | 450 | 0.5903 | 0.4089 | 0.5570 | 0.6016 | 0.5742 | 0.5516 |
| 270000 | 5 | 360 | 0.5819 | 0.3577 | 0.5523 | 0.5991 | 0.5720 | 0.5313 |

**Table 9: Average member training fitness on the Heart Disease problem (150 iterations).**

| Evals | Team Size | Pop Size | Island | Team | OET1 | OET2 | Island/Team | OET1/Team |
|---|---|---|---|---|---|---|---|---|
| 90000 | 2 | 300 | 0.4795 | 0.4169 | 0.4791 | 0.5044 | 0.5038 | 0.4793 |
| 90000 | 3 | 200 | 0.4745 | 0.4154 | 0.4774 | 0.4902 | 0.4876 | 0.4970 |
| 90000 | 4 | 150 | 0.4697 | 0.3292 | 0.4792 | 0.4903 | 0.4706 | 0.4666 |
| 90000 | 5 | 120 | 0.4729 | 0.2918 | 0.4709 | 0.4868 | 0.4628 | 0.4543 |
| 180000 | 2 | 600 | 0.4700 | 0.4323 | 0.4978 | 0.5098 | 0.4935 | 0.4957 |
| 180000 | 3 | 400 | 0.4727 | 0.3888 | 0.4836 | 0.4993 | 0.4770 | 0.4868 |
| 180000 | 4 | 300 | 0.4676 | 0.3646 | 0.4801 | 0.4839 | 0.4753 | 0.4802 |
| 180000 | 5 | 240 | 0.4686 | 0.3034 | 0.4783 | 0.4803 | 0.4717 | 0.4607 |
| 270000 | 2 | 900 | 0.4712 | 0.3746 | 0.5077 | 0.5106 | 0.4873 | 0.4897 |
| 270000 | 3 | 600 | 0.4732 | 0.4082 | 0.4855 | 0.4954 | 0.4828 | 0.4939 |
| 270000 | 4 | 450 | 0.4738 | 0.3654 | 0.4748 | 0.4861 | 0.4680 | 0.4770 |
| 270000 | 5 | 360 | 0.4727 | 0.3225 | 0.4755 | 0.4894 | 0.4707 | 0.4628 |

**Table 10: Average member testing fitness on the Heart Disease problem (150 iterations).**