Parallel distributed multi-objective fuzzy GBML

Keming Li, Liangyu Che
School of Computer Science and Engineering
Southern University of Science and Technology
Email 11612126@mail.sustc.edu.cn
Email 11612228@mail.sustc.edu.cn

1. Introduction

The basic purpose of this project is to base on [1] to implement a Parallel Distributed Multi-Objective Fuzzy Genetics-Based Machine Learning, which can be used to classify patterns.

In this report, we introduce some primery results and conclusion.

- Implement MOCCA(multi-objective cooperative co-evolutionary algorithm).
- Implement random data dimension division method
- Implement a simple structure paralism GA structure, RING
- **Hybrid Fuzzy GBML**:We used python to build Hybrid Fuzzy GBML and run it successfully.
- **NSGA-II Framework**:We implemented the NSGA-II framework for individual sequencing.
- **PCA**:We implemented PCA to reduce dimension of data.
- **Parallel**:We implemented the island model to enable parallel computation.
- **Group cooperation**: Siyi Ding's group used our algorithm to train the fuzzy classifier as the classifier of his microwave recognition project.
- **Test**:We use Breast_W, wine, sonar, pima, Glass and other data sets for training and testing. The results were close to [1].

2. Background Knowledge

2.1. Evolutionary multi-objective optimization

2.1.1. Evolutionary algorithms.

2.1.2. NSGA-II .

2.2. Fuzzy rule-based classifiers

2.3. Coevolutionary algorithms

Natural evolution is often thought of as a population living in a fixed environment, but this is not totally true. Due to the interaction between different species, changes in one species or group of species will affect other species to varying degrees. In biology, coevolution is used to indicate that species living in an environment try to survive depending one on each other.[3]

Traditional evolutionary algorithms work on a fixed environments. In this case, the fitness of the individual is measured using a predefined fitness function. After several iterations, the evolutionary algorithm makes the individuals adapt to the environment according to the established fitness function.

When the scale of the problem is too large, the traditional evolutionary algorithm will consume a large amount of hash power. At the same time, the whole system is inefficient and the results of each generation have low convergence.

The basic idea of coevolutionary algorithms is divide and conquer, dividing a large problem into multiple interdependent subproblems called species. coevolutionary algorithms can be cooperative or competititive as well as interpopulation or intrapopulation. In coevolutionary algorithms, the evolving populations by cooperating with each other for the best fitness, individuals in the same species compete with each other to accelerate the convergence.

coevolutionary algorithms has been also used to solve fuzzy rule classification problems.In the Michigan-style GBML framework, the rules are usually divided into distinct species according to the consequent.

Multi-objective cooperative coevolutionary algorithm is one of cooperative coevolutionary algorithms (CCA).

3. brief introduction of the original design

3.1. MOCCA

There are five issues in MOCCA which are elaborated in this section:

- Problem Decomposition.
- Species Initialization
- Collaboration Formation.
- Fitness Evaluation.
- Framework.

3.1.1. Problem Decomposition. Since our project is based on Pittsburgh-style GBML framework, the problem could

not be directly decomposed according to the consequence of rules.

Our design method is similarly decompose the problem according to the consequence of rules. However, the individual is not a rule but a rule set in which the rules have the same consequence.

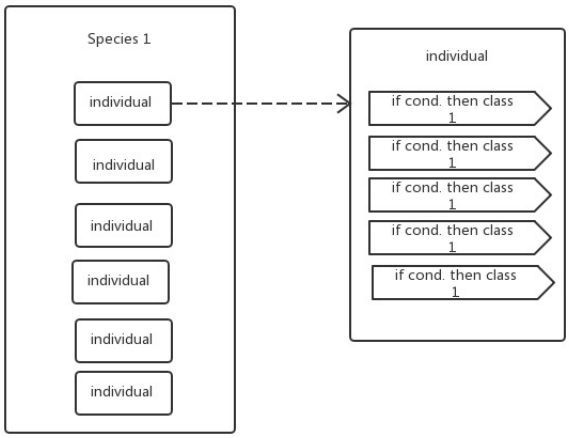

Figure 1. species & individual

3.1.2. Species Initialization. we generate fuzzy rules with the training set, and the rules were then put into the their corresponding rule pools according to the consequence. Next, we randomly select M rules from one rule pool to generate an individual and put it into the corresponding species.

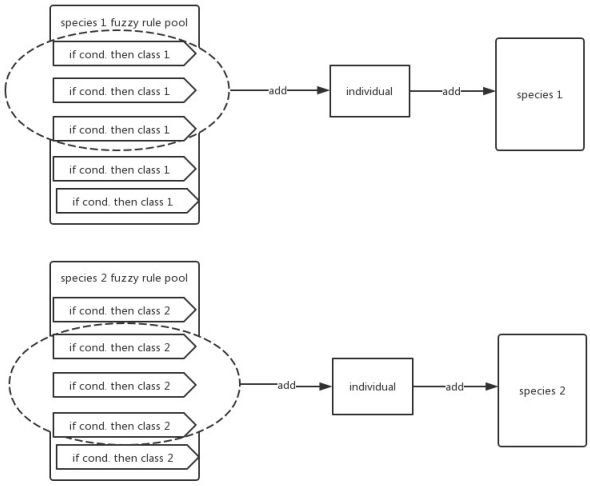

Figure 2. initialization

3.1.3. Collaboration Formation. The individuals in each species are not complete fuzzy classifier and cannot be evaluated directly. Therefore, in order to calculate the fitness of each individual, we need different species to cooperate. Forming collaboration requires the following two steps:

- Representative selection
- Combination operations

Before we evaluate individuals' fitness, we need to select the best individual and two random individuals from each species as representatives of the species. This operation is called representative selection.

An individual will be merged with the optimal represen-

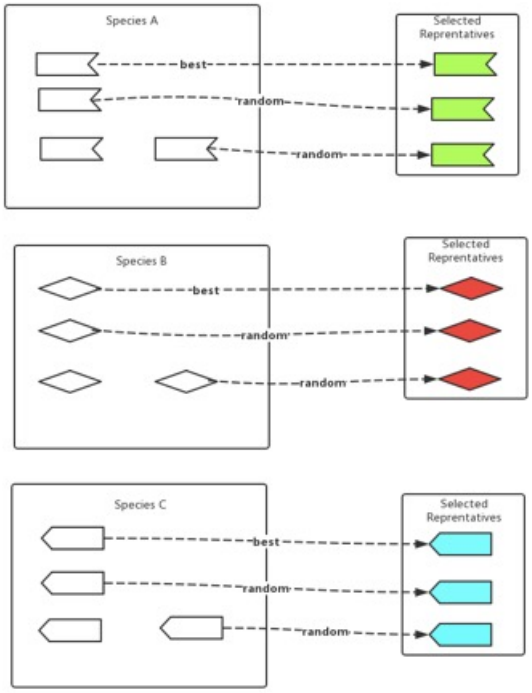

Figure 3. Representative selection

tative of other species to create a fuzzy classifier, and then two fuzzy classifiers will be created by merging the same method with the random representative of other species. This operation is called combination operations.

3.1.4. Fitness Evaluation. We calculate the fitness of the three fuzzy classifiers generated by Combination operations with the fitness function. Then, we select the lowest fitness as the fitness of the individual.

3.1.5. Framework. After initialized species, each species will evolve into offspring through Pittsburgh-style GBML. Then, we obtain the fitness of each individual through species collaboration. Next, enter the next evolution cycle

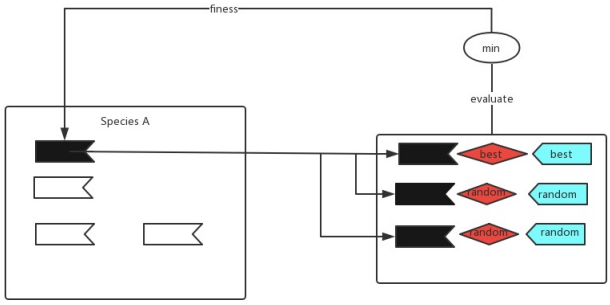

Figure 4. Combination operation

until the end of evolution. Finally, we output the fuzzy classifier which composed of the best individuals in each species.

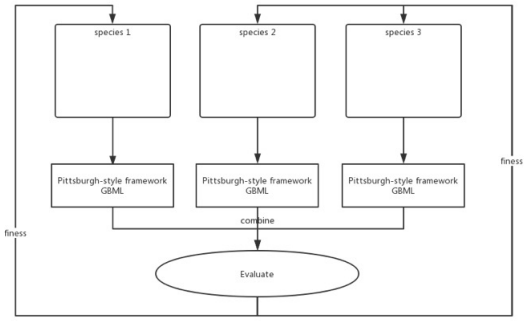

Figure 5. Framework

3.1.6. Pseudocode.

Algorithm 1 MOCCA

Input: $train\_data$
Output: $fuzzy\_classifier$

Species_list ← Initial(train_data)
while gen < N_gen **do**
  for Species ∈ Species_list **do**
    SortWithFitness(Species)
    Species ← Species[:N]
    representative_list ← representative_select(Speices)
    while offspring.length < N **do**
      parent ← Binary_select(Species)
      child ← Crossover(parent)
      mutate(childs)
      offspring ← child
    end while
    Species ← offspring
  end for
  for Species ∈ Species_list **do**
    for individual ∈ Species_list **do**
      classifier ← Combine(individual, representative_list)
      individual.Fitness ← fitness_func(classifier,train_data)
    end for
  end for
end while
fuzzy_classifier ← best_combine(Species_list)
return $fuzzy\_classifier$

4. Experimental results

4.1. Decomposition Strategy

The following data were measured on a supercomputer.
Select wine_change and Glass as data sets
Label decomposition: each subpopulation size is 30, run 150 generations.
Dimensional decomposition: each subpopulation size is 100, run 150 generations.

The tables below compare the performance of the two algorithms in wine_change and glass

algorithm	avg generation time	total time
Label decomposition	12.32s	1903.5s
Dimension decomposition	0.87s	139.2s

TABLE 1. PERFORMANCE IN WINE CHANGE

algorithm	avg generation time	total time
Label decomposition	4.67s	724.6s
Dimension decomposition	1.49s	236.8s

TABLE 2. PERFORMANCE IN GLASS

The following figures show the time spent per generation in the wine_change and glass and the process communication time.

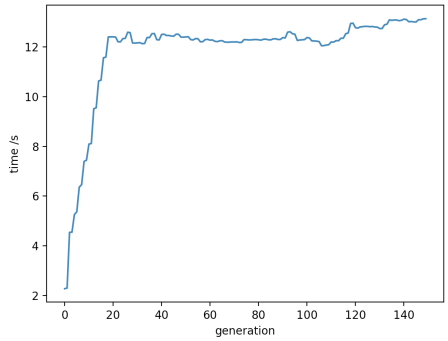

Figure 6. label_decomposition_wine_generation_time

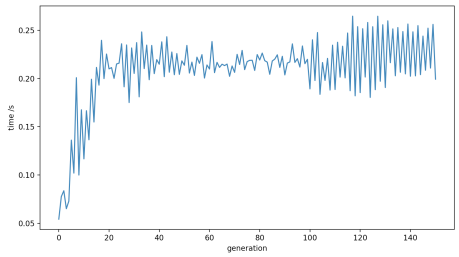

Figure 7. label_decomposition_wine_communication_time

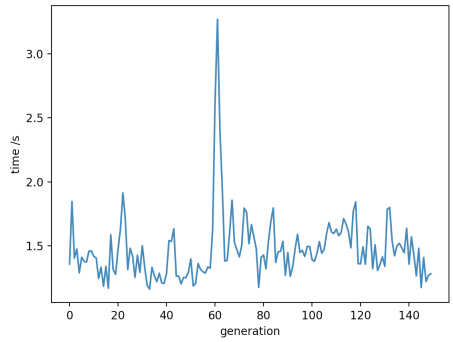

Figure 8. dim_decomposition_wine_generation_time

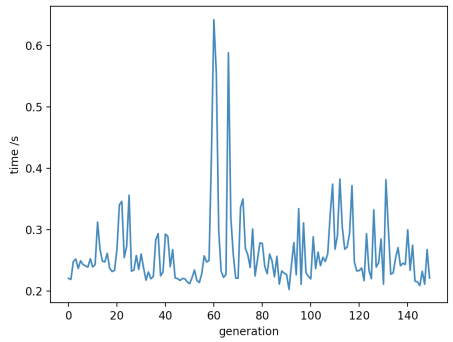

Figure 9. dim_decomposition_wine_communication_time

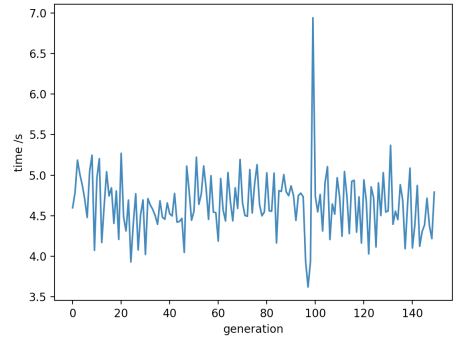

Figure 10. label_decomposition_glass_generation_time

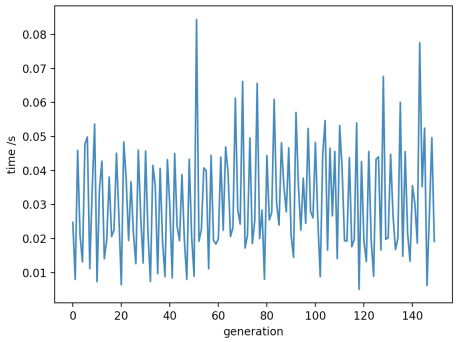

Figure 11. label_decomposition_glass_communication_time

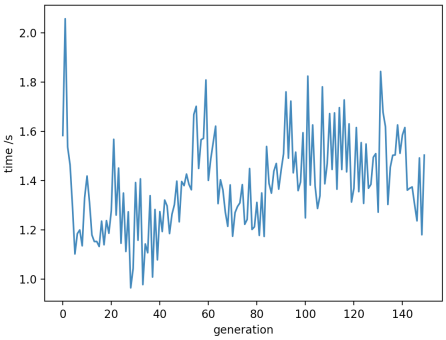

Figure 12. dim_decomposition_glass_generation_time

5. Analysis of experimental results and hypothesis

5.1. Decomposition Strategy

5.1.1. Comparison of run time of different algorithms. When using label decomposition, the individual of a single subpopulation is not a complete classifier, and the classifier generated by combining the subpopulation needs to be reevaluated in each generation, which takes a lot of time.

However, for dimension decomposition, the individual of each subpopulation is the classifier that only classify the specific dimensions (the other dimensions are don't care). In

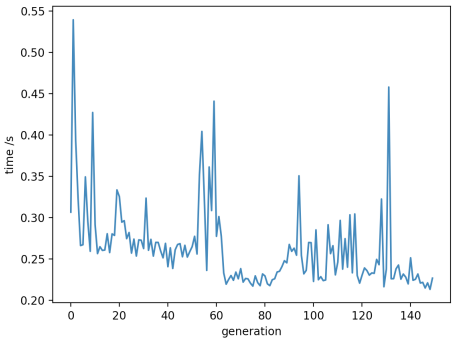

Figure 13. dim_decomposition_glass_communication_time

this way, when cooperating, it only needs to recombine the subpopulations within the main process in ten generation, which consumes less time.

5.1.2. Comparison of run time in different datasets. For label decompositon, the algorithm run faster in the multi-label dataset than in the small-label dataset when those datasets size and dimension are equal.

References

[1] H. Ishibuchi and Y. Nojima, "Analysis of interpretability-accuracy tradeoff of fuzzy systems by multiobjective fuzzy genetics-based machine learning," International Journal of Approximate Reasoning, vol. 44, no. 1, pp. 4-31, January 2007.

[2] X. Yao, Evolving Artificial Neural Networks in Proceedings of the IEEE, vol. 87, no. 9, pp. 1423–1447,September 1999

[3] Xing Zong-Yi, Hou Yuan-Long, Zhang Yong, Jia Li-Min and Hou Yuexian, "A Multi-objective Cooperative Coevolutionary Algorithm for Constructing Accurate and Interpretable Fuzzy systems," 2006 IEEE International Conference on Fuzzy Systems, Vancouver, BC, 2006, pp. 1404-1410.