**Challenges faced in SO Modeling**

1. Large computational time for simulation model
2. **Large number of decision variables**
3. Optimal Solution for Pareto Front

Among these issues we targeted the problem of large number of decision variables

**Different types of categories the optimization algorithm can be divided into based on technical characteristics used:**

1. Multiobjective evolutionary algorithms (MOEAs) based on cooperative coevolution (CC)
2. MOEAs based on decision variable analysis
3. MOEAs based on problem transformation
4. A new search method based on learning strategies

**MOEAs based on cooperative coevolution (CC)**

This method divides the decision variables into multiple groups and then optimizes each group separately.

**MOEAs based on decision variable analysis**

This method focuses on proposing a mechanism for analyzing decision variables and strives to get the best grouping of decision variables. Different from the first category, it is divided into different groups according to the types of decision variables and optimizes for each type of decision variable by different strategies.

The algorithm divides decision variables into distance variables and diverse variables by analyzing the relationship between decision variables and convergence and diversity attributes. The algorithm first optimizes the distance variable and then optimizes the diversity variable.

First, the decision variable clustering method divides decision variables into two categories: convergence-related variables and diversity-related variables. Then, the convergence optimization strategy and the diversity optimization strategy are used to optimize the two types of decision variables.

**MOEAs based on problem transformation**

This method solves the problem by transforming the original large-scale problem into a small-scale problem through the problem transformation function. For example, a weight optimization framework (WOF) was proposed, which optimizes the weight vector instead of decision variables, thus transforming the original large-scale problem into a small-scale problem. Subsequently, a problem reconstruction framework (LSMOF) was proposed, which reconstructs the decision space through a series of reference solutions and weight variables. Then, the original large-scale MOP is transformed into a low-dimensional single-objective problem

**A new search method based on learning strategies**

This method uses the learning mechanism between particles in the original decision space to improve the searchability of the algorithm. Common particle learning mechanisms are particle swarm optimization (PSO), and the competitive swarm optimizer (CSO). The most representative algorithm in this category is LMOCSO. In the algorithm, the inferior particles learn from the superior particles to produce promising offspring, thereby accelerating the global optimization search.

**PROBLEMS:**

Although the existing large-scale MOEAs have shown encouraging performance, each algorithm has its shortcomings.

* MOEAs based on the cooperative coevolution and grouping mechanism need to spend a lot of time analyzing decision variables to complete the grouping of decision variables. In addition, the performance of MOEAs based on the CC framework can be severely degraded due to inappropriate grouping.
* MOEAs based on problem transformation need to find a problem transformation function to ensure that the information loss is as little as possible after the original problem is transformed into a new problem. However, it is very difficult to find a perfect problem transformation function, and it is even more impossible in particularly complex problems.
* MOEAs based on learning strategies directly find the optimal solution in the original decision space. In a decision space of hundreds of dimensions, this type of algorithm can find a better solution by increasing the number of function evaluations of the evolutionary algorithm. As the dimensionality of decision variables increases, the size of the decision space increases exponentially. In MOPs with thousands of dimensional decision variables, it is not enough to solve the problem simply by increasing the number of function evaluations of the evolutionary algorithm.

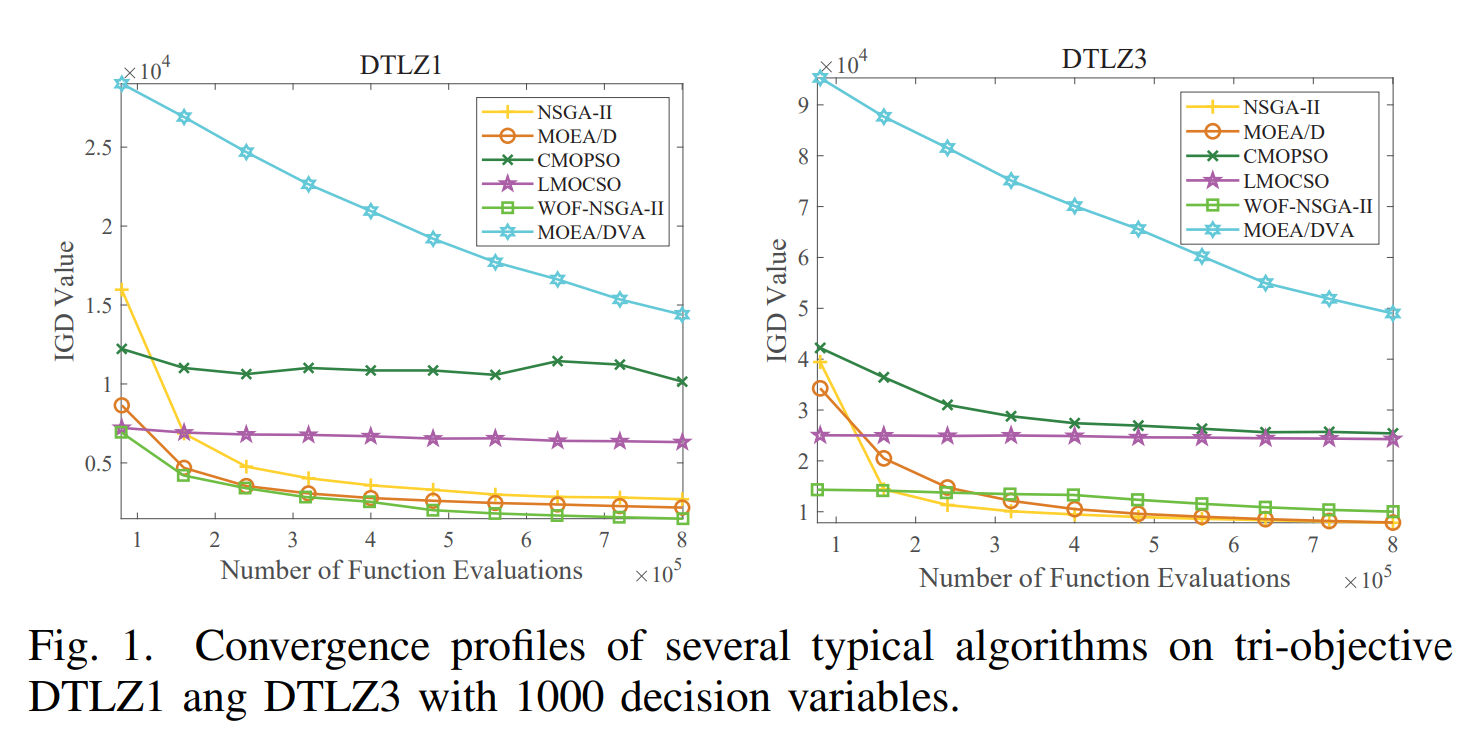
Therefore, a fuzzy decision variables (FDV) framework is proposed for large-scale multiobjective optimization.

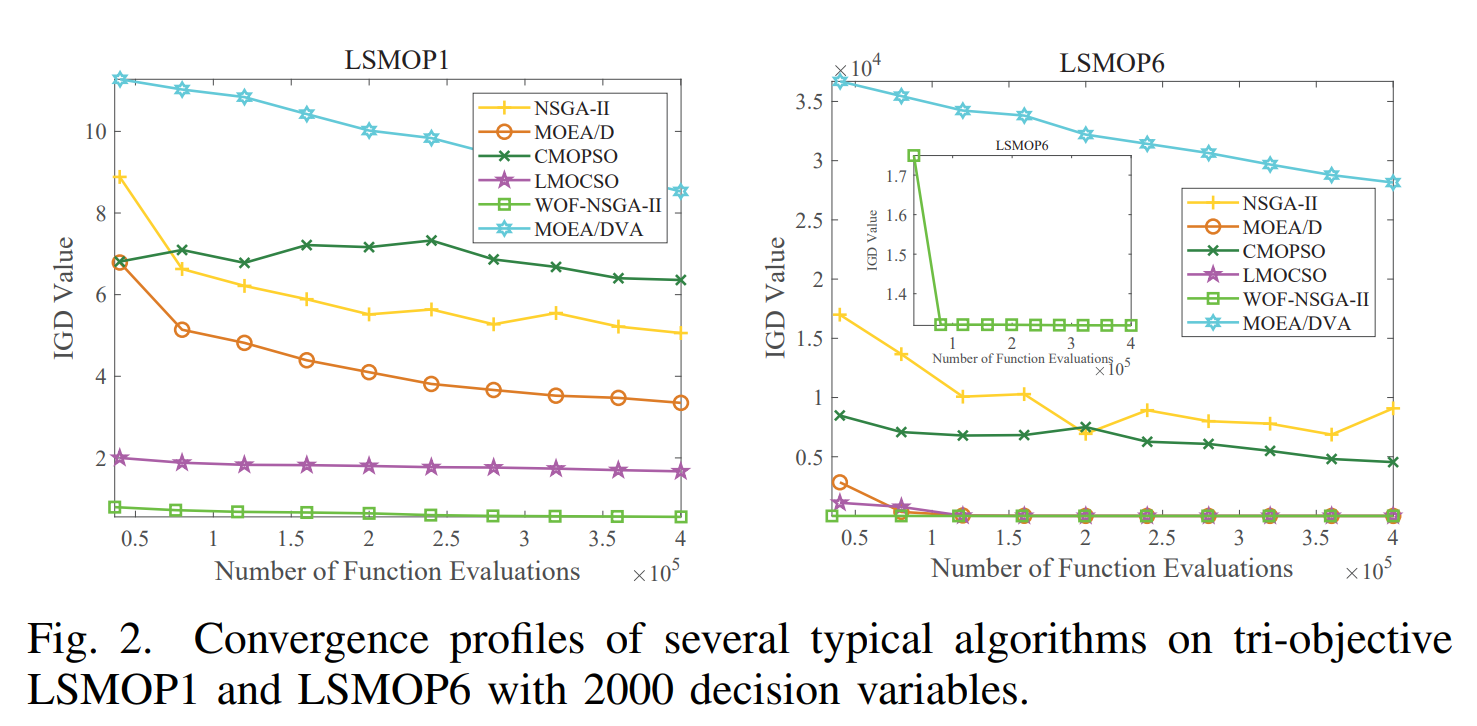
**Fuzzy Decision Variables(FDV) Framework**

* A method of fuzzy evolution sub-stages division is proposed. This method divides the fuzzy evolution stage into multiple sub-stages with a gradually decreasing degree of fuzzification. The higher the degree of fuzzification in the sub-stage, the lower the accuracy of the solution obtained by FDV optimization
* Formulas for fuzzifying decision variables are proposed. The fuzzy formula adapts to multiple fuzzy evolution sub-stages. In a sub-stage, first, the two fuzzy target values of the decision variables are calculated. Then the degree of membership is calculated for the decision variable belonging to two fuzzy sets, and a fuzzy set is mapped to a fuzzy target value. Finally, the value of the decision variable is fuzzified into a fuzzy target value represented by a fuzzy set with a larger degree of membership.
* In order to verify the effectiveness of the proposed FDV in solving MOPs, several representative MOEAs are embedded in the FDV and compared with the original algorithm on the multiobjective test suite DTLZ. Experimental results show that MOEAs embedded in the FDV are significantly better than the original algorithm in all test cases, and the possibility of using existing MOEAs to solve large-scale problems is realized. Furthermore, the CSO is embedded in the FDV framework (FDVCSO) and then compared with several state-of-the-art large-scale MOEAs on the large-scale multiobjective test suite LSMOP. Experimental results show that FDVCSO is significantly better than other large-scale MOEAs in most test cases.

**Performance of existing large-scale MOEAs in MOPs with thousands of dimensional decision variables**

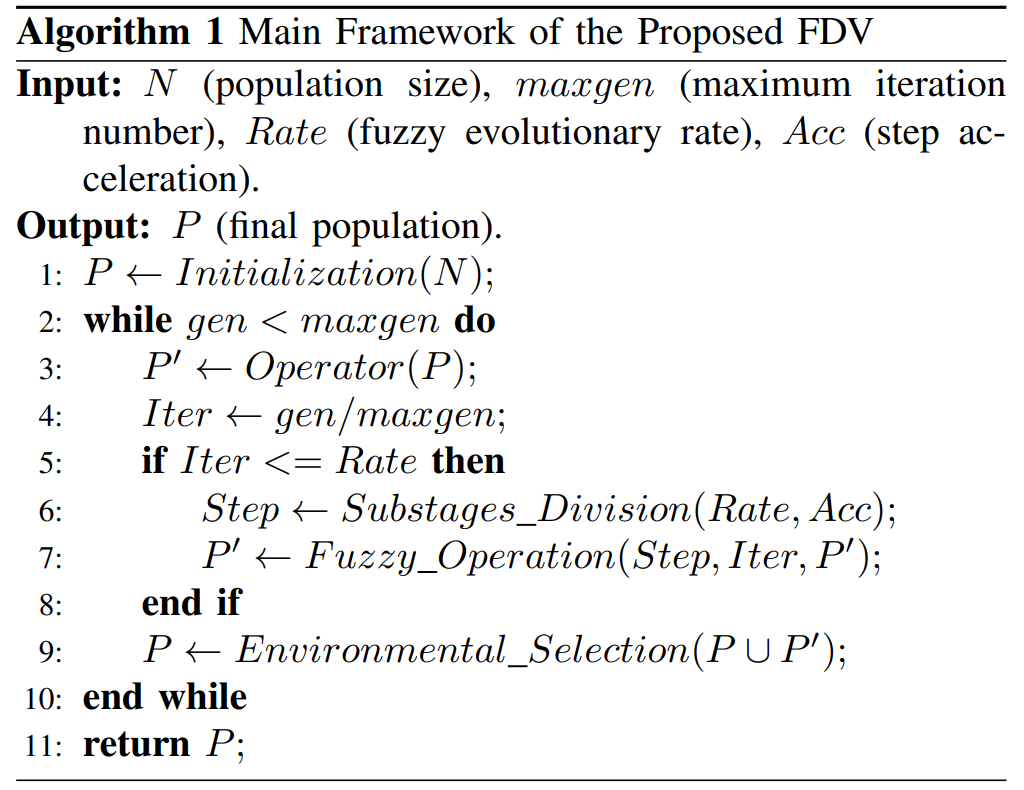
The performance of existing large-scale MOEAs in MOPs with thousands of dimensional decision variables is unsatisfactory, especially the convergence. For example, using NSGAII, MOEA/D, CMOPSO, LMOCSO, WOF-NSGA-II, and MOEA/DVA, experiments have been conducted on the DTLZ1 and DTLZ3 problems with tri-objective and 1000 decision variables. The plot of the convergence profiles of the mean IGD values achieved by these algorithms is displayed in Fig. 1. Fig. 1 shows that large-scale MOEAs (LMOCSO, WOFNSGA-II, and MOEA/DVA) did not perform better than classic MOEAs (NSGA-II, MOEA/D, and CMOPSO), and even performed worse than classic MOEAs. Using MOEA/DVA, WOF-NSGA-II, LSMOF, and LMOCSO, experiments have been conducted on the LSMOP1 and LSMOP6 problems with tri-objective and 2000 decision variables. The plot of the convergence profiles of the mean IGD values achieved by these algorithms is displayed in Fig. 2. Fig. 2 shows that the existing large-scale MOEAs also converged poorly on largescale testing problems.





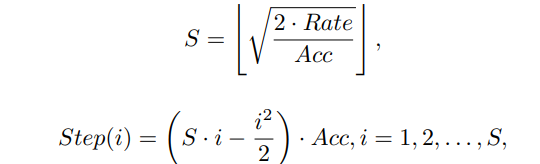
**PROPOSED FRAMEWORK OF FDV**

The main scheme of the proposed FDV is presented in Algorithm 1. First, the population P is initialized by the population initialization method of the embedded MOEA. gen is the current number of iterations. The offspring P’ are generated by the offspring generation method of the embedded MOEA. Iter is the ratio of the current number of iterations to the total number of iterations. Then, the whole evolution process is divided into two stages: fuzzy evolution stage and precise evolution stage. The first stage is to add fuzzy evolution sub-stages division (line 6) and fuzzy operation (line 7) after the offspring generation operation. In the second stage, only offspring generation operations are performed. Finally, the next-generation population is screened through the embedded MOEA environmental selection method. It is worth noting that the precise evolution stage can directly evolve through the embedded MOEA. The fuzzy evolution stage is composed of two main components: fuzzy evolution sub-stages division and fuzzy operation.



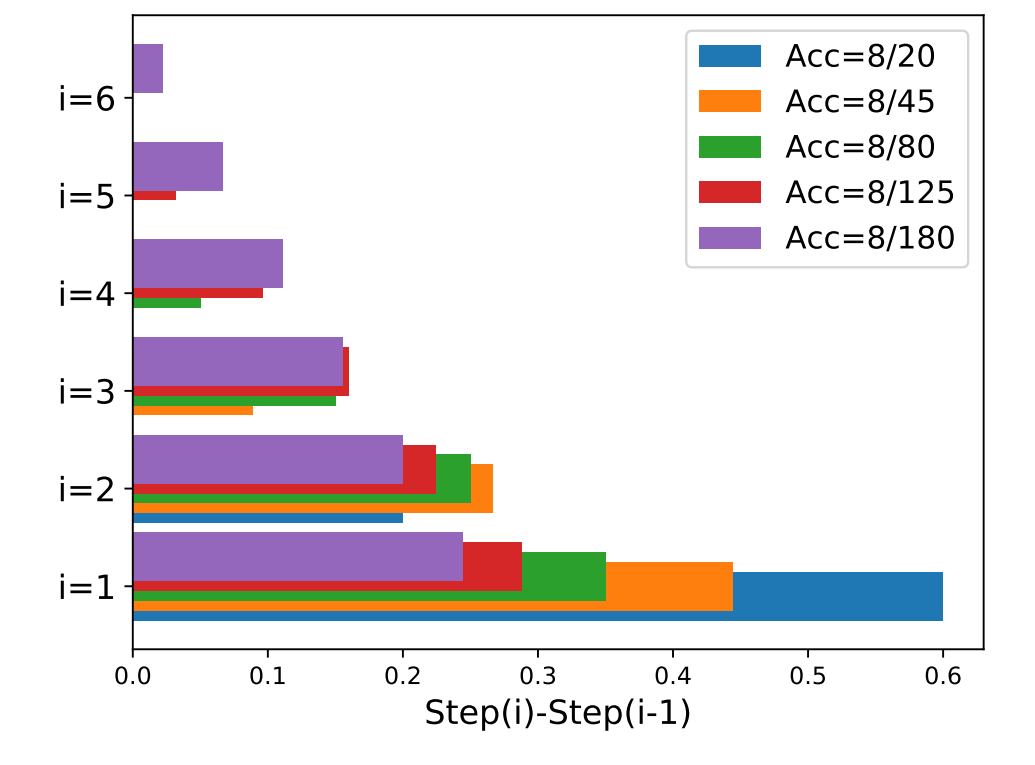
**A. Fuzzy Evolution Sub-stages Division**

The purpose of dividing the entire fuzzy evolution stage into multiple fuzzy sub-stages with decreasing degrees of fuzzification is to make the solution obtained by the algorithm more and more accurate. The mathematical description of the fuzzy evolutionary sub-stages division method is as follows:



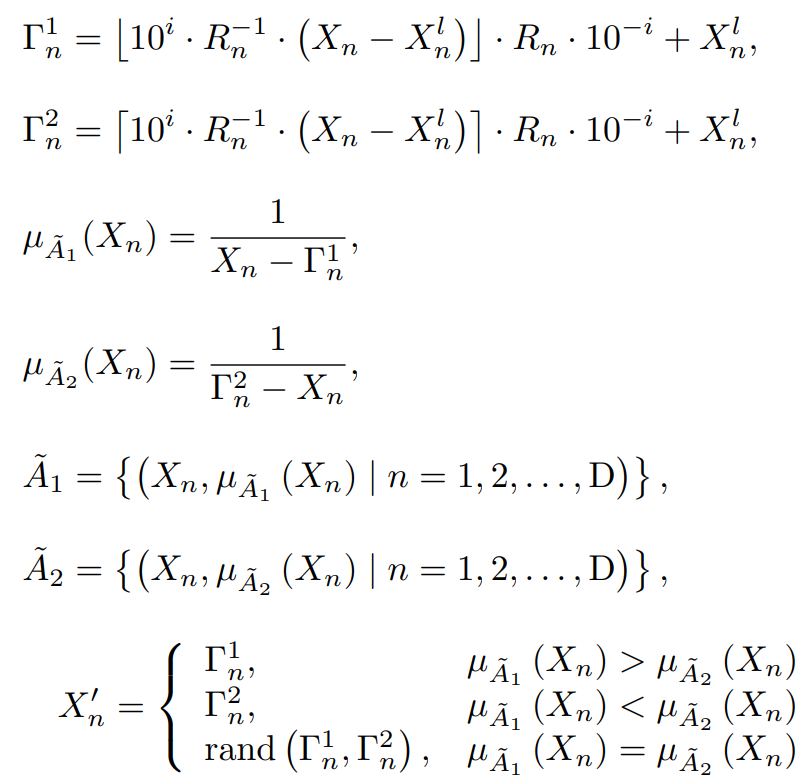
where S represents the number of sub-stages divided in the fuzzy evolution stage; Step(i) is the cumulative step length of the first i sub-stages in the fuzzy evolution stage, the default Step(0) = 0; Total = 1 represents the total step length of the entire evolution process; Rate represents the proportion of fuzzy evolution stage in the entire evolution process, namely, fuzzy evolution rate; Acc > 0 is the step acceleration, which can control the step length change speed between sub-stages; notably when Step(S) < Rate, Step (S + 1) = Rate is added.

Step(i) − Step(i − 1) is the step length of the i-th fuzzy evolution sub-stage. In Fig. 4, the earlier fuzzy evolutionary sub-stages have a longer step length of evolution, so sub-stages that have a higher degree of fuzzification last longer time in the evolutionary process. When the value of Acc increases under the premise that Rate does not change, the fuzzy evolution is divided into fewer sub-stages. On the contrary, the fuzzy evolution is divided into more sub-stages. When FDV has poor convergence in the early stage of evolution, a larger Acc should be set to keep the algorithm in the high-fuzzification stage for a longer time. If it is found that the FDV appears to have prematurely converged during the optimization process, It may be that the algorithm has fallen into a local optimum. Then the Acc should be reduced to increase the search time for searching for low-fuzzy and high-precision solutions. If it is found that the final result obtained by the algorithm is not ideal, the Rate of FDV should be increased appropriately. In general, in the face of different problems, different Acc and Rate parameters should be set.

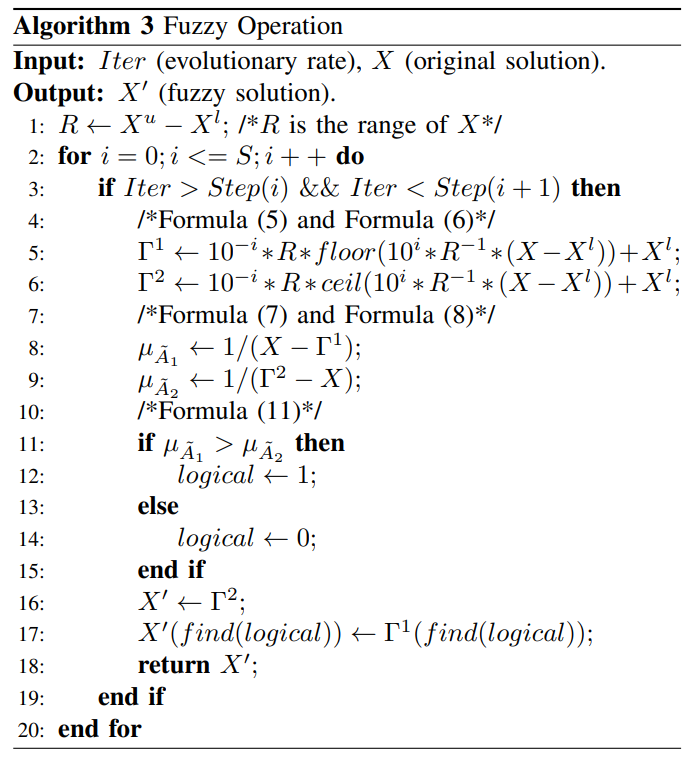
`

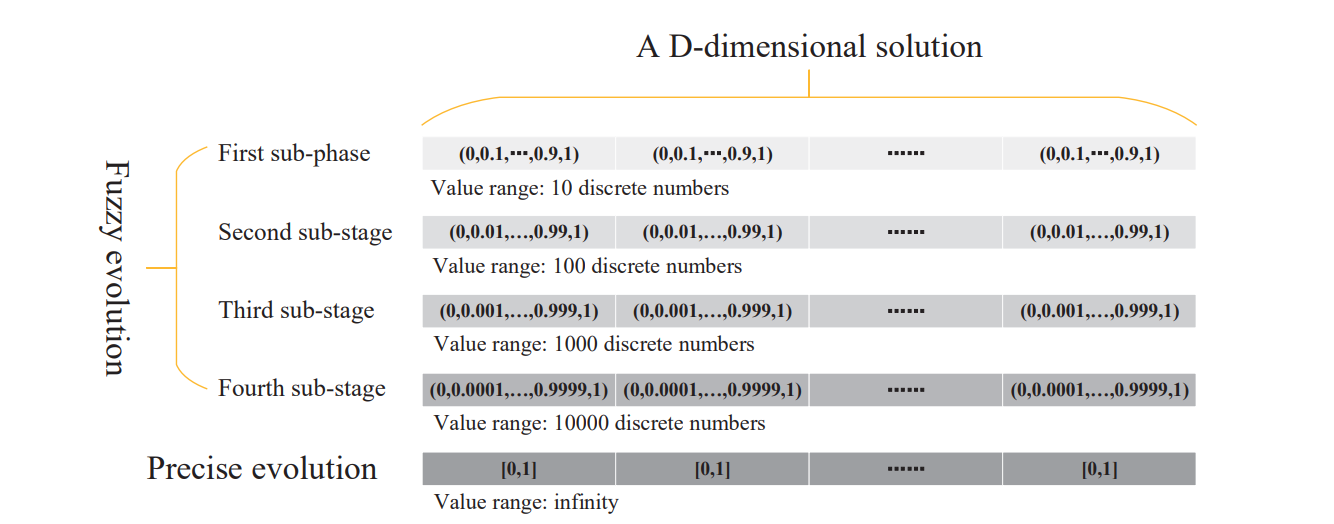
**B. Fuzzy Operation**

The entire fuzzy evolution stage is divided into multiple fuzzy evolution sub-stages with decreasing degrees of fuzzification. Therefore, how to determine the degree of fuzzification in a fuzzy evolution sub-stage and how an original solution is fuzzified will be introduced in this section. The knowledge of fuzzy sets and membership functions used in fuzzification is described in Section II. This paper proposes a series of formulas for fuzzy decision variables as follows:



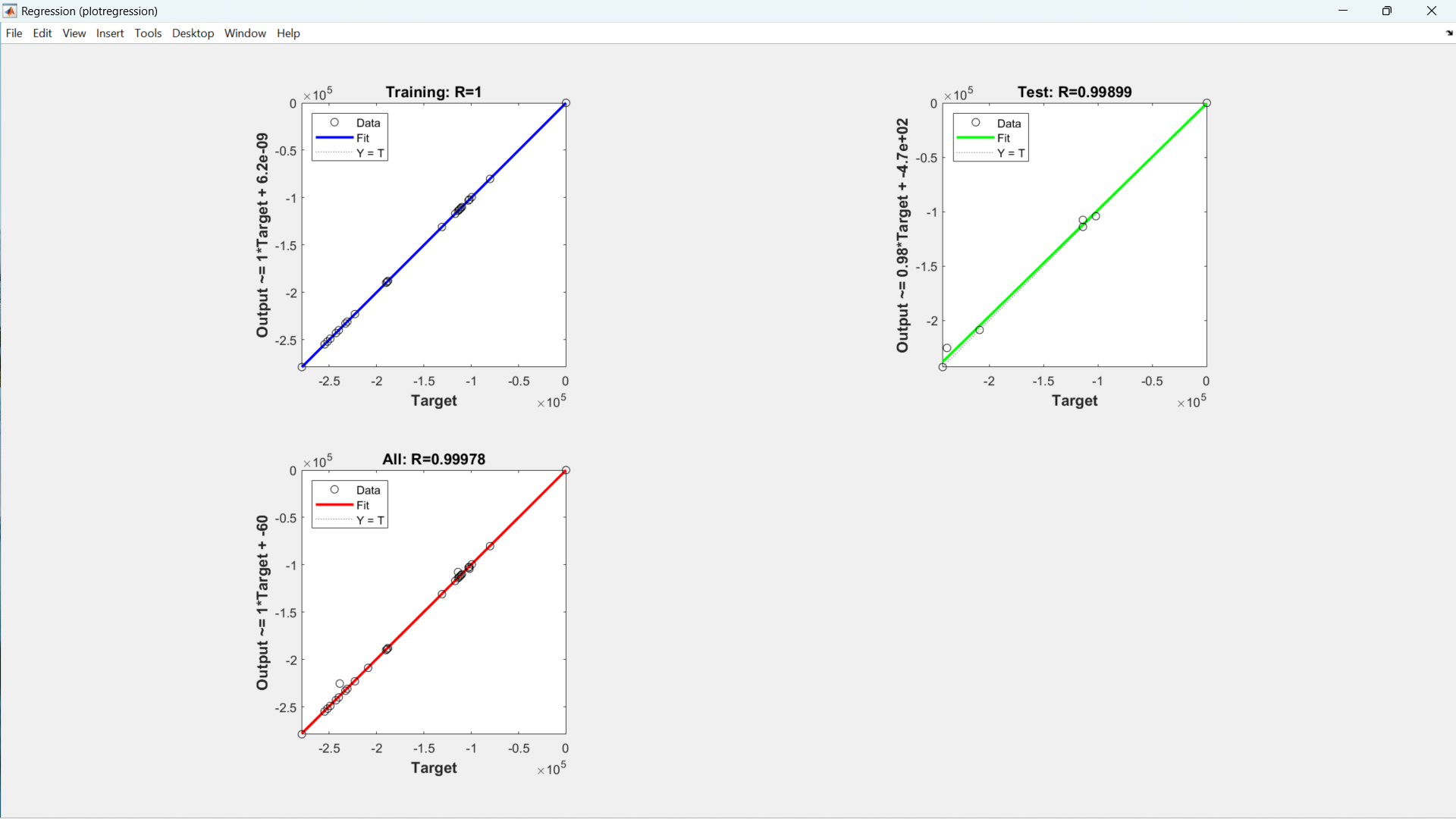
ip. Although fuzzy operation involves related knowledge of fuzzy theory, its implementation process is simple and easy to understand. The procedure of the fuzzy operation is given in Algorithm 3. The operations in Algorithm 3 are all matrix calculations. First, obtain the length of the decision variable value interval. Judge the current fuzzy evolution sub-stages (line2 and line3) according to the evolution rate Iter. Next, calculate the two fuzzy target values Γ 1 and Γ 1 of the decision variables. Γ 1 and Γ 2 correspond to fuzzy sets Ae1 and Ae2 respectively. Calculate the membership degree of decision variables in two fuzzy sets. Then, update the value of the decision variable according to the degree of membership. The update rule is that the value of the decision variable will be updated to the fuzzy target value corresponding to the fuzzy set with a larger membership degree. Finally, The program returns to the fuzzy solution. In particular, logical is a boolean variable matrix, f ind(1) returns true, and find(0) returns false.





**ANN TRAINING**

We have trained Neural Network model for the dataset. ANN training result on small data:



But, In order to train our model even better, we have taken data of **20020** samples with 10 hidden layers, we got even better results.

Error histogram:

