A GA based knowledge discovery system for the design of fluid dispensing processes for electronic packaging

K.Y. Chan and C.K. Kwong[1]

Department of Industrial and Systems Engineering,

The Hong Kong Polytechnic University,

Hung Hom, Kowloon,

Hong Kong

Abstract: In the semiconductor manufacturing industry, fluid dispensing is a very common process used for die-bonding and microchip encapsulation in electronics packaging. Understanding the process behaviour is important as it aids in determining appropriate settings of the process parameters for a high-yield, low cost and robust operation. In this paper, a genetic algorithm (GA) based knowledge discovery system is proposed to discover knowledge about the fluid dispensing process. This knowledge is expressed in the form of rules derived from experimental data sets. As a result, appropriate parameters can be set which will be more effective with respect to the required quality of encapsulation. Rules generated by the GA based knowledge discovery system have been validated using a computational system for process optimization of fluid dispensing. The results indicate that the rules generated are useful and promising in aiding optimization of the fluid dispensing process in terms of better optimization results and shorter computational time.

Keywords: Genetic algorithm, knowledge discovery system, fluid dispensing process, electronics packaging

[1] Email of the corresponding author: mfckkong@inet.polyu.edu.hk

1 Introduction

Fluid dispensing process is a popular way to perform microchip encapsulation for chip-on-board packages [15]. To study the process, it is common for engineers to conduct a large number of experiments and generate experimental data sets. Experimental data sets must first be processed and/or analyzed in order to extract patterns, useful information or knowledge. The development of effective and efficient methods for deriving knowledge from these data is important as the knowledge extracted from the data not only has a high predictive accuracy but also is comprehensible by users [11,12,13]. Recent developments of data mining algorithms, such as Bayesian networks [5], nonlinear regression and classification methods [9], example based methods [22], frequent patterns [34], decision or regression trees [30], relational learning models [7] and evolution algorithms [1,2] have drawn considerable attention from academics and from industry.

Rule induction is one of the common forms of data mining [25]. It is a method for discovering a set of "IF THEN" rules that can be used for converting uninformative data into either a knowledge base for decision support or an easily understood description of the system behavior so that knowledge that humans can understand can be explored. It also features the capability to search for all possible interesting patterns from data sets. The neural network approach has been employed to extract rules for solving crisp and fuzzy classification problems [14,19,27]. However, neural networks do not supply any analytical guidance for determining the network configuration. A network may also be trapped into local optima in the learning process. This problem puts a limitation on the quality of rules generated from neural networks. Also traditional rule induction methods discover the rules only by

adopting local heuristic search techniques [11] that are also likely to be trapped into local optima.

Recently the approach of using genetic algorithms to induct rules from data bases has been found useful due to its simplicity and its capability as a powerful search mechanism that [20] evaluates the rule as a whole through the fitness function, rather than evaluating the impact of adding or removing one condition to or from a rule. Further extension of this can be found in [6,28]. Genetic algorithms use a population of individual solution structures called chromosomes. Theory shows that the knowledge about desirable solutions is advantageously stored in the population itself. Such knowledge is implicitly contained in the surviving chromosomes through evolution [16]. The principle of inducting rules, essentially using the genetic algorithm, can be taken as the core of the method. It can also be found from recent literature that genetic algorithms have been applied in mining rules from chemical data [32], financial data [33], manufacturing data [26], qualitative bankruptcy data [21], zoo data [7], nursery data [7], heart disease data [4].

In this paper, a knowledge discovery system based on a genetic algorithm (GA) for mining rules from a number of experimental data sets concerned with the fluid dispensing process, is proposed. Currently, engineers determine the process parameters to select the settings in fluid dispensing, by using their experience and intuitive judgments. This leads to them spending a long time in determining the proper settings. With the use of the rules generated from the knowledge discovery system, it is hoped that the time of identifying proper process parameters setting can be significantly reduced. The organization of this paper is as follows: Section 2 introduces the fluid dispensing for microchip encapsulation in electronic packaging. Section 3 presents the operations of the proposed GA based knowledge discovery

system for rule mining. In Section 4, validation of the rules generated by the GA based knowledge discovery system is performed with the aid of the developed computational system [24]. Numerical results and discussion are also given. Finally, a conclusion is drawn.

2 Fluid dispensing for microchip encapsulation

In fluid dispensing processes of microchip encapsulation, normally, silicon chips are covered with an epoxy encapsulant using an X-Y numerically controlled dispensing system that delivers the epoxy encapsulant through a needle. The material is commonly dispensed in a pattern, working from the center outwards. An epoxy dam around the die site and second wire bond points can be made to contain the flow of material and this produces a more uniform looking part as shown in Figure 1. Fluid dispensing is a highly nonlinear process and creates a highly coupled multi-variable system that involves complex inter-relationships between the epoxy properties, process conditions, needle design parameters and overall encapsulation quality. In semiconductor manufacturing, trial-and-error is still a common method used to identify appropriate process parameter settings. However, this method involves a long process development time and optimum encapsulation quality may not be obtained. A detailed description of fluid dispensing can be found in [15].

To determine an optimal process condition of fluid dispensing, understanding the process behavior is necessary. With assistance from the supporting company of this research, three significant process parameters and their normal operating ranges were identified as follows:

- The compressed air pressure (1 bar to 4 bar), x_1
- The height between the substrate and the needle (250 to 2000 steps of a

stepping motor), x_2

- The pump motor speed (400 rpm to 1000 rpm), x_3.

Two quality characteristics were studied in this research which are the encapsulation weight (mg), y , and the encapsulation thickness (mm), z. 96 experiments were carried out based on a full factorial design with 4 levels in compressed air pressure (x_1), 4 levels in the height between the substrate and the needle (x_2) and 6 levels in pump motor speed (x_3).

3 GA based rule discovery system

In this section, a genetic algorithm GA based knowledge discovery system of the fluid dispensing for microchip encapsulation, which is used to generate rules from the experimental data sets, is described. First, an experimental data set, involving process parameters and measures of encapsulation, are collected by carrying out experiments on the fluid dispensing process. Then a knowledge discovery system that consists of a conjunction of encapsulation requirements and the rules consequently recommended for searching domains of process parameters, is developed by the genetic algorithm. Based on the GA based rule discovery system, informative rules involving a small searching domain of process parameters can be recommended with respect to the required encapsulation. The rules generated can be represented as follows.

$$\text{if } y = y_w \text{ and } z = z_w \text{ then } R_1^l \le x_1 \le R_1^u \text{ and } R_2^l \le x_2 \le R_2^u \text{ and } R_3^l \le x_3 \le R_3^u$$

where y_w is the required encapsulation weight; z_w is the required encapsulation thickness; $R_1^l \le x_1 \le R_1^u$ is the range of setting of the process parameter x_1 ; $R_2^l \le x_2 \le R_2^u$ is the range of setting of the process parameter x_2; $R_3^l \le x_3 \le R_3^u$ is the range of setting of the process parameter x_3. All the ranges are recommended by the

GA based knowledge discovery system. With a set of training data samples, Figure 2 shows a schematic diagram of the GA based knowledge discovery system.

Details of the GA based knowledge discovery system are described below:

3.1 Generation of random strings

The first step of the GA based knowledge discovery system is to randomly generate a population of strings which represent the ranges of the process parameters. The strings can be expressed as $\left[R_1^l, R_1^u, R_2^l, R_2^u, R_3^l, R_3^u\right]$, where R_i^l and R_i^u are the lower and upper ranges of the i^{th} process parameter x_i with *i*=1, 2 and 3 respectively.

Real and binary encoding are two commonly used approaches for string representation in GAs. In binary encoding representation, strings need to be encoded to real values for fitness evaluation and also they need to be decoded again for reproduction operations. However, in real encoding representation, there is no need for string encoding and decoding. Leaving out encoding and decoding can help to reduce the computational time. Since the ranges of process parameters are all real values, real encoding is chosen.

3.2 Fitness evaluation

The fitness function of the GA based knowledge discovery system is used to evaluate how good a rule fits the data samples of the epoxy dispensing process. Due to the limited number of data sets, the required conditions of encapsulation weight y_w and thickness z_w are covered by the ranges $Y_w^l \le y_w \le Y_w^u$ and $Z_w^l \le z_w \le Z_w^u$ defined by the following rule:

if $\mathrm{Y}_w^l \le y = y_w \le \mathrm{Y}_w^u$ and $\mathrm{Z}_w^l \le z = z_w \le \mathrm{Z}_w^u$ then

$$R_1^l \le x_1 \le R_1^u \text{ and } R_2^l \le x_2 \le R_2^u \text{ and } R_3^l \le x_3 \le R_3^u$$

where the ranges $Y_w^l \le y_w \le Y_w^u$ and $Z_w^l \le z_w \le Z_w^u$ covers 10% of the whole operating ranges of the encapsulation weight and encapsulation thickness respectively; and R_1^l, R_1^u, R_2^l, R_2^u R_3^l and R_3^u are the values of the string as discussed in Section 3.1 and they determine the fitness of a rule.

Rules need to be evaluated during the training process in order to establish points of reference for the GA based knowledge discovery system. The fitness function considers the data sets as: correctly classified, left to be classified, and the wrongly classified ones. In the GA based rule discovery system, the fitness function (1), which was suggested by Carvalho and Freitas [2] is used. The fitness function evaluates the predictive accuracy of a rule based on both *true positive rate* and *true negative rate* that considerably mitigates some pitfalls associated with the problems of overfitting and lack of balance,

$$Fitness = true\_positive\_rate \times true\_negative\_rate \tag{1}$$

where $true\_positive\_rate = \dfrac{(\text{no. of } TP)}{(\text{no. of } TP) + (\text{no. of } F\text{N})}$ (2)

and $true\_negative\_rate = \dfrac{(\text{no. of } TN)}{(\text{no. of } TN) + (\text{no. of } F\text{P})}$ (3)

with

- TP means *True Positive* which refers to the data sets covered by the rule correctly classified;
- FP means *False Positive* which refers to the data sets covered by the rule wrongly classified;
- TN means *True Negatives* which refers to the data sets not covered by the rule but differing from the training target class;

- FN means *False Negatives* which refers to the data sets not covered by the rule but matching the training target class.

With the higher numbers of TP and TN, and the lower numbers of FP and FN, a better rule is generated. For a comprehensive discussion about rule-quality measures, the reader can refer to [18].

The following shows a rule generated by the GA knowledge discovery system:

$$\text{if } 65.2 \le y = 67 \le 68.1 \text{ and } 0.55 \le z = 0.59 \le 0.62 \text{ then} \\ 1 \le x_1 \le 2 \text{ and } 50 \le x_2 \le 600 \text{ and } 250 \le x_3 \le 400 \tag{4}$$

where $y = 67$ and $z = 0.59$ are the required values of the encapsulation weight and encapsulation thickness respectively; $R_1^l(=1)$, $R_1^u(=2)$, $R_2^l(=50)$, $R_2^u(=600)$, $R_3^l(=250)$ and $R_3^u(=400)$ are the values from the string of the GA based knowledge discovery system. To evaluate the fitness of the rule, the 4 training data sets as shown in Table 1 are used,

Classifications of the training data sets are shown in the last column of Table 1.

- The 1-st data set is classified as FN class, since *y*=70.1 is not within the range, $65.2 \le y \le 68.1$, and also both $x_1 = 0.8$ and $x_3 = 200$ are not within the ranges, $1 \le x_1 \le 2$ and $250 \le x_3 \le 400$. This means the sample is not covered by the rule but matches the rule.
- The 2-nd data set is classified as FP class, as *y*=64.3 and *z*=0.51 are not within the ranges $65.2 \le y \le 68.1$ and $0.55 \le z \le 0.62$ respectively, but all $x_1 = 1.2$, $x_2 = 400$ and $x_3 = 350$ are within the ranges, $1 \le x_1 \le 2$, $50 \le x_2 \le 600$ and $250 \le x_3 \le 400$. Therefore the data set is not covered by the rule but is wrongly classified as belonging to the target class.

- The 3-rd data set is classified as TP class, since *y*=66.9 and *z*=0.57 are all within the ranges $65.2 \le y \le 68.1$ and $0.55 \le z \le 0.62$ respectively, and also all $x_1 = 1.8$, $x_2 = 350$ and $x_3 = 300$ are within the ranges $1 \le x_1 \le 2$, $50 \le x_2 \le 600$ and $250 \le x_3 \le 400$ respectively. Therefore the data set is covered by the rule and is correctly classified.
- The 4-th data set is classified as TN class, since *y*=65.5 and *z*=0.61 are all within the ranges $65.2 \le y \le 68.1$ and $0.55 \le z \le 0.62$ respectively, and both $x_2 = 40$ and $x_3 = 220$ are not within the ranges $50 \le x_2 \le 600$ and $250 \le x_3 \le 400$ respectively. This means the data set is not covered by the rule but differs from the target class.

In this example, the number of data sets in all FN, FP, TP and TN classes is 1. Thus based on the fitness function (1), the fitness of rule (4) can be calculated as:

$$Fitness = true\_positive\_rate \times true\_negative\_rate$$
$$= \frac{TP}{TP+FN} \times \frac{TN}{TP+FP} = \frac{1}{1+1} \times \frac{1}{1+1} = 0.25$$

3.3 Convergence and Selection

The population is evolved and improved in each generation until a stopping condition is met. In genetic algorithms, there are quite a few stopping conditions. In this research, the stopping criterion is fulfilled when the number of generations is equal to a pre-defined number of generations or one of the solutions in the population of the genetic algorithm achieves a full fitness score of 1. Otherwise the GA based knowledge discovery system performs the selection operation for the next evolutionary generation.

For the selection of strings, the approach of the roulette-wheel is used, which is one of the most common selection methods used for selecting strings to perform reproduction operations [16]. This is unlike other selection approaches such as rank based selection [35], tournament selection [17], where their selective pressures need to be controlled by adjusting their inbuilt parameters. The selection of strings produced by the roulette-wheel selection algorithm is completely based on the fitness of the strings. Therefore it can provide a zero bias to strings in the population.

This selection method imitates the roulette-wheel game, where the dice thrown would most probably end up by being in the slot with the largest area. Following this, one can conclude that the string with the largest fitness value is most likely to be chosen because it has the largest slot size. The fitness value of the j^{th} string in a population is fit_j. The fitness values are used to calculate the probability of selection, $prob_j$, to the j^{th} string. The probability of selection $prob_j$ is defined as:

$$prob_j = \frac{fit_j}{\sum_{j=1}^{Popsize} fit_j} \tag{5}$$

where *Popsize* is the population size of the GA based knowledge discovery system.

3.4 Crossover and mutation

Discrete Crossover Operation [29] is the most common crossover operation and is performed by exchanging variable values between parent strings. However, it can only generate corners of the hypercube defined by the parent strings. Furthermore, experimental results have indicated that the combination of biases is far from optimal and has undesirable side-effects on the exploratory power of crossover [10]. Another common crossover operation for real encoding representation is Intermediate

Crossover [29]. It is capable of producing any point within a hypercube which is larger than that defined by the parent strings. Therefore it can be adapted to sustain a higher explorative search in the searching domain than when using Discrete Crossover Operation.

In the development of the GA based knowledge discovery system, intermediate crossover, which can produce a new string around and between the variables of the two selected parent strings, is used. Referring to the representation of the genetic algorithm, a new string $\left[R_1^{\,l}, R_1^{\,u}, R_2^{\,l}, R_2^{\,u}, R_3^{\,l}, R_3^{\,u}\right]$ is produced according to the following rule:

$$\begin{aligned}\left[R_1^{\,l}, R_1^{\,u}, R_2^{\,l}, R_2^{\,u}, R_3^{\,l}, R_3^{\,u}\right] &= \left[{}^1R_1^{\,l}, {}^1R_1^{\,u}, {}^1R_2^{\,l}, {}^1R_2^{\,u}, {}^1R_3^{\,l}, {}^1R_3^{\,u}\right] + \\ \alpha\left\{\left[{}^1R_1^{\,l}, {}^1R_1^{\,u}, {}^1R_2^{\,l}, {}^1R_2^{\,u}, {}^1R_3^{\,l}, {}^1R_3^{\,u}\right] - \left[{}^2R_1^{\,l}, {}^2R_1^{\,u}, {}^2R_2^{\,l}, {}^2R_2^{\,u}, {}^2R_3^{\,l}, {}^2R_3^{\,u}\right]\right\}\end{aligned} \qquad (6)$$

where α is a scaling factor chosen uniformly at random over an interval $\left[-0.25, 1.25\right]$, and $\left[{}^1R_1^{\,l}, {}^1R_1^{\,u}, {}^1R_2^{\,l}, {}^1R_2^{\,u}, {}^1R_3^{\,l}, {}^1R_3^{\,u}\right]$ and $\left[{}^2R_1^{\,l}, {}^2R_1^{\,u}, {}^2R_2^{\,l}, {}^2R_2^{\,u}, {}^2R_3^{\,l}, {}^2R_3^{\,u}\right]$ are the two selected parent strings. Ranges of process parameters in the new string are the result of combining the values of the parent strings according to (6) with a scaling factor α chosen for each range of process parameter. In geometric terms, intermediate crossover is capable of producing new parameter values within a slightly larger hypercube than that defined by the parent strings, but these values are constrained by a range of scaling factor α.

Mutation is carried out by randomly changing one or more values of a selected string between the operating ranges of process parameters. During mutation, the value of each range of process parameter in a rule has a finite probability of changing. Therefore the probability of searching within the operating ranges of process parameters is never zero. This prevents complete loss of genetic material through selection and elimination.

The mutation operator of Gaussian perturbation [29] of individual variables was used in the GA based knowledge discovery system. For example, the variable R_j is selected to be mutated. After performing the mutation, its value becomes:

$$R_j' = R_j + \text{MutMx} \times D_j \times \delta \quad (7)$$

where MutMx = +1 or -1 with equal probability; D_j = 0.5 × operating range of the *j*-th process parameter; δ = a value in the range [0,1] for shrinking the mutation range based on Gaussian perturbation.

After being generated the newly produced strings are put into the old population to generate a new population. This can be done by replacing the least fit strings in the old population with the newly produced strings. Such replacement can also be produced by randomly replacing the strings in the old population with the newly produced strings. In this research, a random reinserting approach was used.

3.4 Rule Induction

96 experiments were carried out based on a full factorial design with 4 levels in compressed air pressure (x_1), 6 levels in pump motor speed (x_2) and 4 levels in the height between the substrate and the needle (x_3). 88 out of the 96 experimental data sets were used to train the GA based knowledge discovery system, and the remaining 8 experimental data sets were used for system validation.

The GA based knowledge discovery system was implemented using Matlab programming software. The parameter settings, Crossover rate = 0.8 and Mutation rate =$1/n$, where n is the number of variables of the string, suggested by [31] were adopted. Since the number of variables of the string is 6 (i.e. *n*=6), mutation rate was

set at $1/6$. The number of generations and population size were set at 500 and 100 respectively.

If the required encapsulation weight *y* is 50 mg and the required encapsulation thickness *z* is 0.5 mm, Figure 3 shows a rule recommended by the GA based knowledge discovery system. From Figure 3, it can be found that the numbers of TP = 6, FP = 1, FN = 4, TN =77, and the fitness value of the recommended rule is 0.5923. From the rule, more specified ranges of parameter settings can be obtained.

4 Results Verification

To validate the effectiveness of the rules generated by the GA based knowledge discovery system the computational system for fluid dispensing developed by Kwong et al [24] was employed.

Given operating ranges of process parameters (x_1, x_2, x_3), and the required encapsulation weight β and thickness γ, the computational system determines the setting of the three process parameters, compressed air pressure (x_1), pump motor speed (x_2), and the distance between the substrate and needle (x_3), based on the requirements of encapsulation weight β and thickness γ. The system consists of a neural network (NN) based prediction model, and a GA based optimization unit as shown in Figure 4. Here we call it a pure computational system.

In the GA based optimization unit, the following objective function is used:

$$\text{Objective Function: } Min\left(\lambda_1 \frac{|y-\beta|}{\beta} + \lambda_2 \frac{|z-\gamma|}{\gamma}\right) \tag{8}$$

subject to: $1 \le x_1 \le 4$, $400 \le x_2 \le 1000$, $250 \le x_3 \le 2000$,

where λ_1 and λ_2 are the weights of the two quality characteristics, encapsulation weight and encapsulation thickness respectively.

To validate the effectiveness of the GA based knowledge discovery system, the system was integrated with the pure computation system developed by Kwong et al [24] as shown in Figure 5. In the enhanced computational system, recommended ranges of parameter settings are generated by the GA based knowledge discovery system and input to the GA based optimization unit. It is hoped that the parameter settings recommended by the enhanced computational system will lead to better results of the two quality characteristics than the pure computational system.

To validate the GA based knowledge discovery system, eight validation tests were carried out. First, eight sets of required encapsulation weights and thicknesses as shown in Table 2 were inputted to the GA based knowledge discovery system.

The corresponding eight rules were generated as shown below:

1 IF y=72.3 and z =0.58

THEN 1.6124<x1<2.4414 AND 403<x2<1682.7 AND 493.99<x3<687.3

2 IF y=43.2 and z =0.48

THEN 2.5476<x1<3.268 AND 250<x2<2000 AND 850<x3<1000

3 IF y=87.4 and z =0.67

THEN 1.0561<x1<2.718 AND 644.34<x2<1862.7 AND 400<x3<541.85

4 IF y=37.2 and z =0.46

THEN 1<x1<1.4294 AND 1665.5<x2<2000 AND 819.61<x3<1000

5 IF y=75.1 and z =0.62

THEN 1<x1<2.5259 AND 960.94<x2<1497.2 AND 400<x3<537.3

6 IF y=59.3 and z =0.57

THEN 1.8869<x1<2.8044 AND 250<x2<465.58 AND 548.05<x3<745.13

7 IF y=62.4 and z =0.53

THEN 1.2048<x1<3.9475 AND 250<x2<897.88 AND 773.09<x3<834.33

8 IF y=53.1 and z =0.53

THEN 1.103<x1<2.3456 AND 250<x2<2000 AND 764.92<x3<849.78

Those recommended ranges of parameters setting were then input to the GA based optimization unit in order to reduce the searching space. Because both the pure computational system and the enhanced computational system involve the stochastic algorithm GA, 50 runs were carried out in the eight validations. Then we evaluated the effectiveness and robustness of both systems by analyzing the statistical results of the 50 runs.

The search results of optimizing the relative errors of both encapsulation weight and encapsulation thickness of the pure computational system and the enhanced computational system are shown in Figure 6 – Figure 21. In the figures, the *x*-axis and the y-axis show the generation numbers and the relative error of the encapsulation respectively. It can be observed clearly from all the figures that in general, the convergence speeds of the enhanced computational system are faster than those based on the pure computational system. Also the relative errors of prediction of the enhanced computational system are all smaller than those of the pure computational systems in all the validation tests.

To investigate the quality and the robustness of solutions found in both the pure computational system and the enhanced computational system, the means and the variances of the relative errors found in both systems for the 8 validations were analysed. Table 3 and Table 4 show the means and variances of the relative errors of both systems respectively. It can be seen clearly from the tables that the enhanced computational system can yield better solutions in terms of mean errors and variance of relative errors compared with the pure computational system in the 8 validations.

The *t*-test was introduced to evaluate the significance between the pure computational system and the enhanced computational system. Table 5 shows two sets of all *t*-values between the pure computational system and the enhanced

computational system for the validation tests for both encapsulation weight and encapsulation thickness. It can be found that all the *t*-values are higher than 2.15, which indicates that the significance is 98% level of confidence. Therefore the performance of the enhanced computational system is significantly better than the pure computational system with 98% confidence, in terms of prediction accuracy.

5 Conclusion

In this paper, a GA based knowledge discovery system was proposed and developed to generate rules from experimental data sets of the fluid dispensing process in which three process parameters, compressed air pressure, the height between the substrate and the needle and pump motor speed, and two quality requirement, encapsulation weight and thickness, are involved. Based on rules generated from the GA based knowledge discovery system, more specified ranges of process parameter settings can be obtained. Engineers could make use of the specified ranges to shorten their time in determining the appropriate setting of process parameters for fluid dispensing compared with the time they spent on their conventional practice. To validate the effectiveness of the rules generated from the GA based knowledge discovery system, the system was integrated with a computational system for fluid dispensing developed by Kwong et al. [24]. Eight validation tests were carried out. Results of the tests indicate that the enhanced computational system can recommend process parameter settings which lead to smaller prediction errors as well as variance of errors in comparison with the Kwong's computational system [24]. Actual experiments will be performed to further verify and validate the effectiveness of the GA based knowledge discovery system.

Further study will involve improving the accuracy and on shortening the computational time of the GA based knowledge discovery system by incorporating a statistical method, orthogonal design, into the GA. The resulting system will be applied on mining useful rules on the survey data sets for car door design.

Acknowledgements

The work described in this paper was supported substantially by a grant from the Hong Kong Polytechnic University.

References

1. Bojarczuk C.E., Lopes H.K. and Freitas A.A. (1999). A hybrid genetic algorithm/decision tree approach for coping with unbalanced classes. Proceedings of the 3rd International Conference Practical Applications of Knowledge Discovery and Data Mining (pp. 61-70).
2. Carvalho D.R. and Freitas A.A. (2000). A hybrid decision tree/genetic algorithm for coping with the problem of small disjoints in data mining. Proceeding of the Conference of Genetic and Evolutionary Computation (pp. 1061-1068).
3. Chipperfield A.J. and Fleming P.J. (1995). The MATLAB genetic algorithm toolbox. Proceedings of the IEE Colloquium on Applied Control Techniques using MATLAB (pp. 10/1-10/4).
4. Chiu C. and Hsu P.L. (2005). A constraint-based genetic algorithm approach for mining classification rules. IEEE Transactions on Systems, Man, and Cybernetics –Part C: Applications and Reviews, 35(2), 205-220.

5. Cooper G.F. and Herskovits E. (1991). A Bayesian method for constructing Bayesian belief networks from databases. Proceedings of the 7th Annul Conference of Uncertainty in Artificial Intelligence (pp. 86-94).

6. Davis L. (1991). Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York.

7. Dehuri S. and Mall R. (2006). Predictive and comprehensible rule discovery using a multi-objective genetic algorithm. Knowledge-based Systems, 19, 413-421.

8. Dzeroski S. (1996). Inductive logic programming and knowledge discovery in databases. Advances in Knowledge Discovery and Data Mining, AAAI Press (pp. 117-152), Menol Park, CA.

9. Elder J. and Pregibon D. (1996). A statistical perspective on knowledge discovery in database. Advances in Knowledge Discovery and Data Mining, AAAI Press (pp. 83-113) Menlo Park, Calif.

10. Eshelmann L.J. (1991). The CHC Adaptive Algorithm: How to have safe search when engaging in non-traditional genetic recombination. Foundations of Genetic Algorithms 1 (pp. 265-283).

11. Fayyad U., Piatetsky-Shapiro G., and Smyth P. (1996). From data mining to knowledge discovery: An overview, Advances in Knowledge Discovery and Data Mining (pp. 1-36).

12. Freitas A.A. (1997). A genetic programming framework for two data mining tasks: classification and generalized rule induction. Proceedings of 2nd Annual Conference on Genetic Programming (pp. 96-101).

13. Freitas A.A. (1999). On rule interestingness measures. Knowledge-Based Systems, 12, 309-315.

14. Giles C., Lee C., Lawrence S. and Tsoi A.C. (1997). Rule inference for financial prediction using recurrent neural networks. IEEE Proceedings of Conference on Computational Intelligence for Financial Engineering (pp. 253–259).
15. Gilleo, K. (2004). Area array packaging processes. NewYork:McGraw-Hill.
16. Goldberg D.E. (1989). Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley.
17. Goldberg D.E. and Deb K. (1991). A comparative analysis of selection schemes used in genetic algorithms. Foundations of Genetic Algorithms (pp. 69-93).
18. Hand D.J. (2001), *Principles of Data Mining*, MIT Press.
19. Hayashi Y. and Imura A. (1990). Fuzzy neural expert system with automated extraction of fuzzy if–then rules from a trained neural network. The First International Symposium on Uncertainty Modeling and Analysis (pp. 489–494).
20. Holland J.H. (1975). Adaptation in Natural and Artificial Systems. Ann Arbor, MI: Univ. Michigan Press.
21. Kim M.J. and Han I. (2003). The discovery of experts' decision rules from qualitative bankruptcy data using genetic algorithms. Expert Systems with Applications, 25, 637-646.
22. Kolodner J. (1993). Case-Based Reasoning. Morgan Kaufman, San Francisco.
23. Kwong C.K., Chan K.Y. and Wong H (2007). Empirical approach to modeling fluid dispensing for electronic packaging. International Journal of Advanced Manufacturing Technology, in print.
24. Kwong C.K., Chan. K.Y. and Wong H. (2007). A computational system of process optimization of fluid dispensing for electronic package. (Reviewing)
25. Langlery P. and Simon H. (1995). Application of machine learning and rule induction. Communication of ACM, 38(11) 54-64.

26. Lee, R. and Sikora, M.S. (1995). A genetic algorithm based approach to flexible flow-line scheduling with variable lot sizes. IEEE Transactions on Systems, Man, and Cybernetics – Part B, 27(1) 36–54.

27. Lin, C. T., and Lee, C. S. G. (1991). Neural network-based fuzzy logic control and decision system. IEEE Transactions on Computer, 12, 1320–1336.

28. Michalewicz Z. (1994). Genetic Algorithms + Data Structure = Evolution Programs. Springer-Verlag, Berlin.

29. Muhlenbein H and Voosen D.S. (1993). Predictive models for the breeder genetic algorithm: I. Continuous parameter optimization. Evolutionary Computation. 1(1), 25-49.

30. Quinlan J. (1992). C4.5: Programs for machine learning. Morgan Kaufmann, San Francisco.

31. Schaffer J., Caruana R., Eshelman L. and Das R. (1989). A study of control parameters affecting online performance of genetic algorithms for function optimization. Proceedings of the 3rd International Conference on Genetic Algorithms (pp. 51-60).

32. Sikora R. (1992). Learning control strategies for a chemical process: A distributed approach. IEEE Expert, 35-43.

33. Sikora R. and Shaw M. (1994). A double-layered learning approach to acquiring rules for classification: Integrating genetic algorithms with similarity-based learning. ORSA Journal on Computing, 6(2), 174–187.

34. Srikant R. and Agrawal R. (1996). Mining sequential patterns: generalizations and performance improvements. Proceedings of the 5th International Conference on Extending Database Technologies (pp. 3-17).

35. Whitley D. (1989). The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. Proceedings of the Third International Conference on Genetic Algorithms (pp. 116-121).

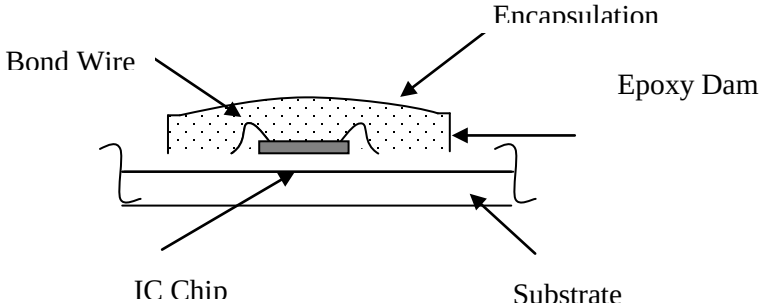

Figure 1 Encapsulation of COB packages

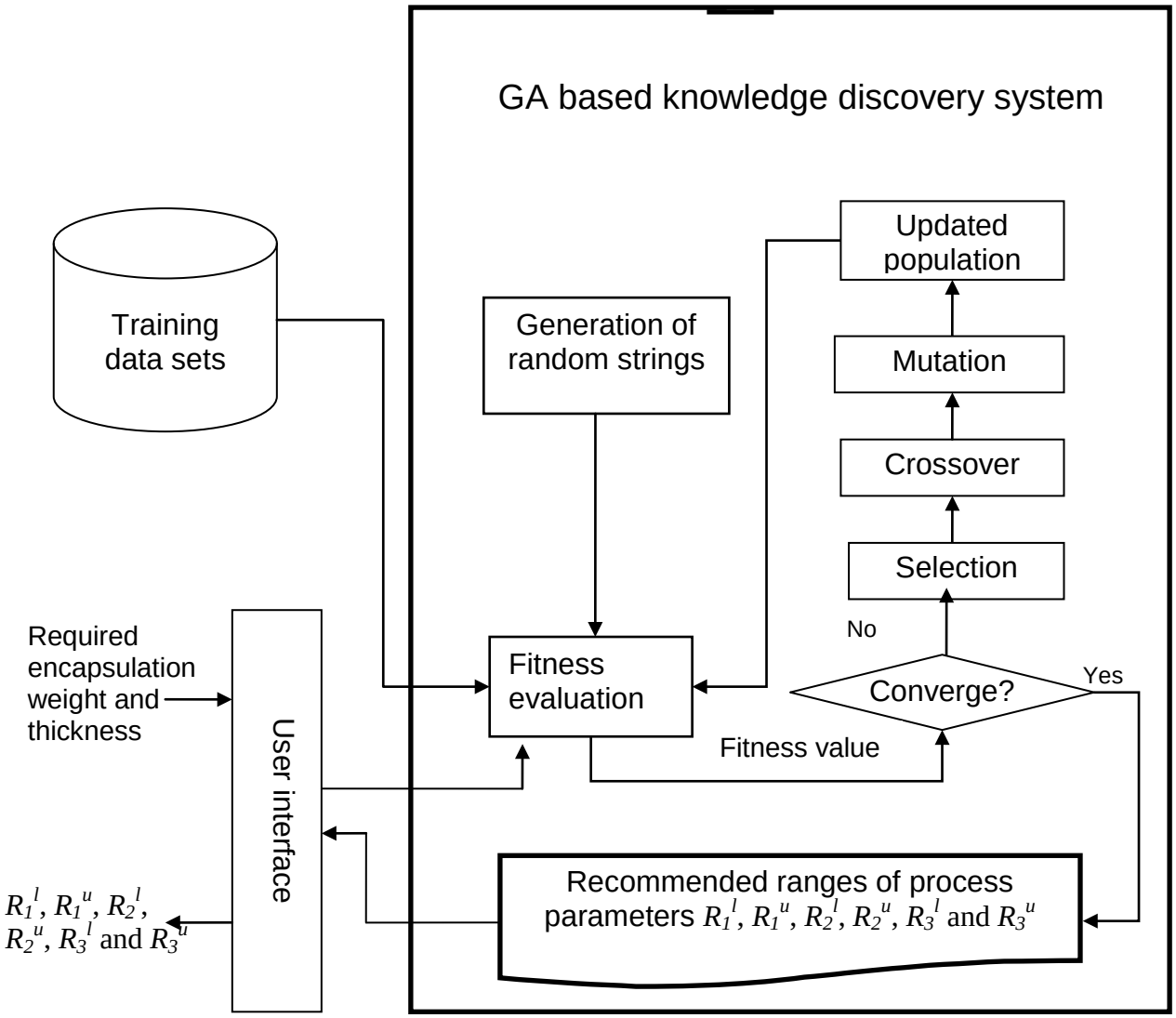

Figure 2 GA based knowledge discovery system

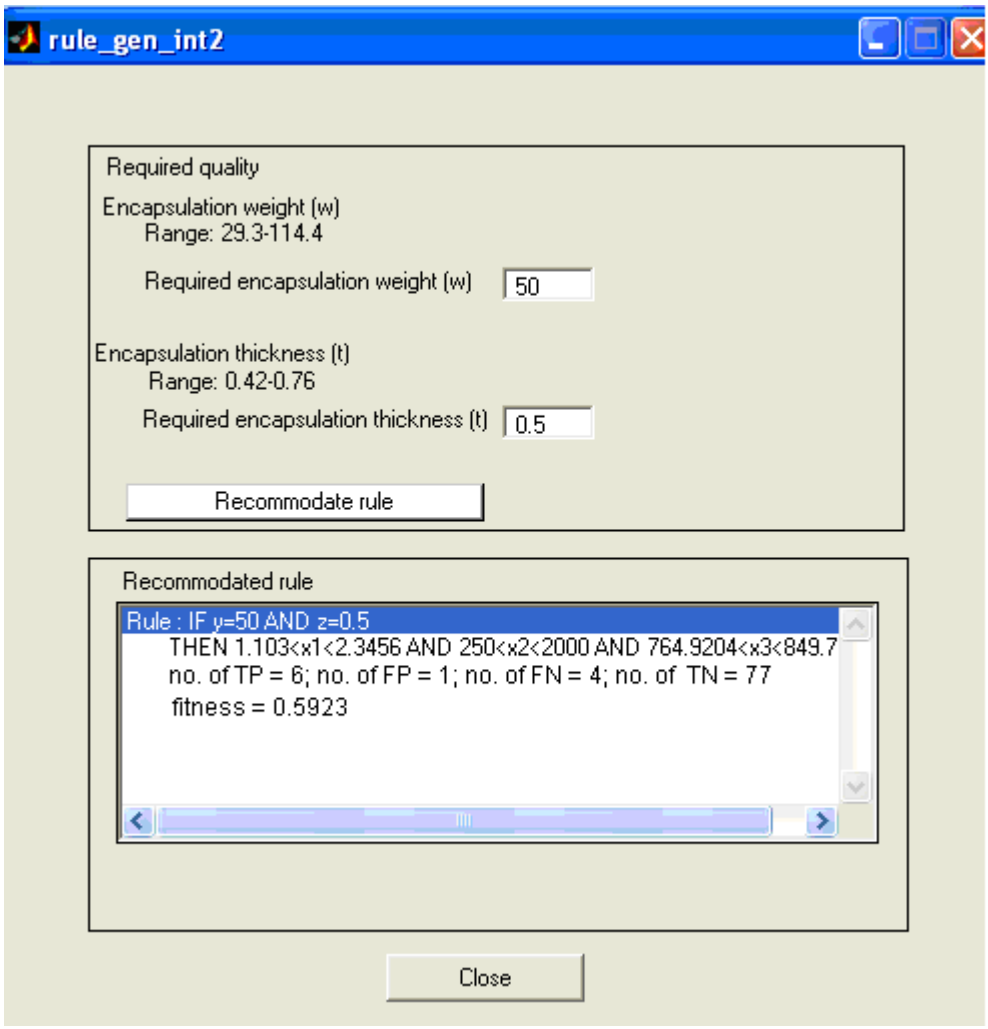

Figure 3 The user interface of the GA based knowledge discovery system

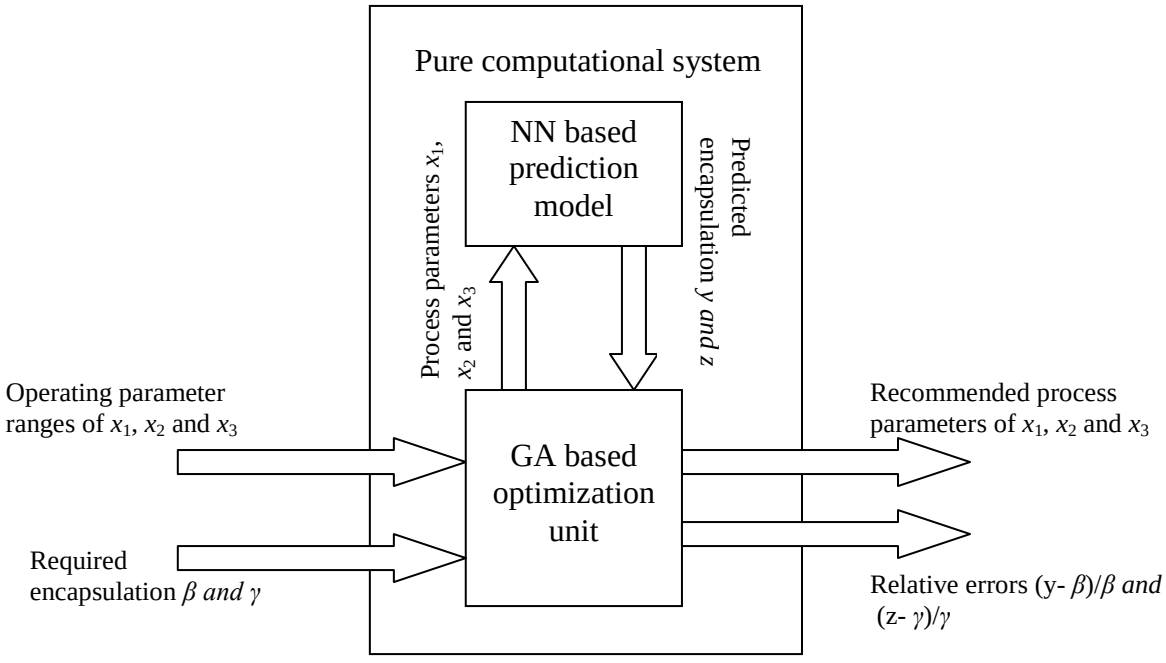

Figure 4 Pure computational system

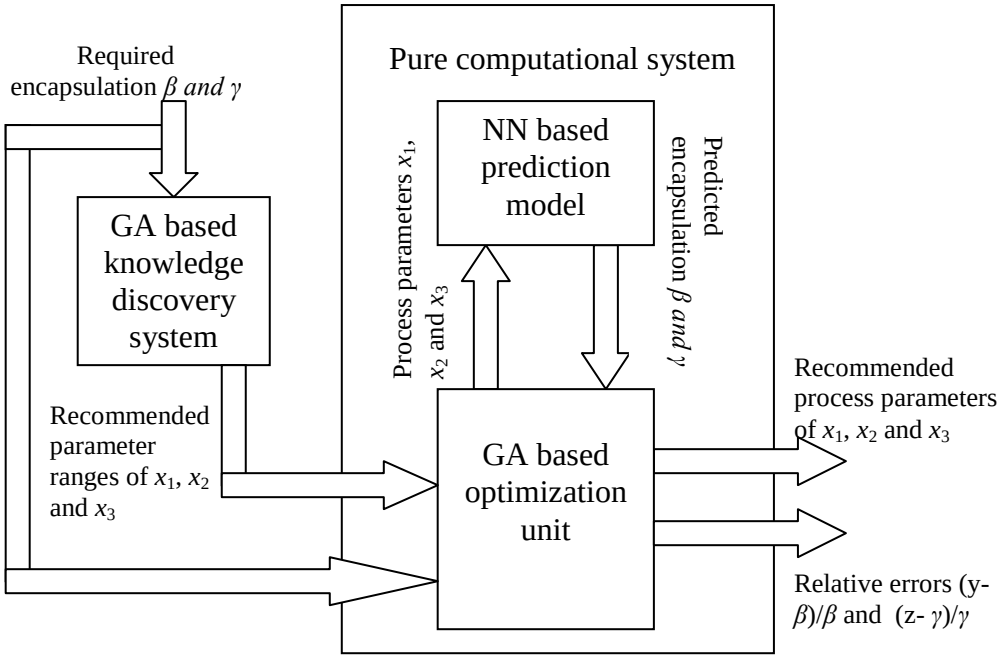

Figure 5 Enhanced computational system

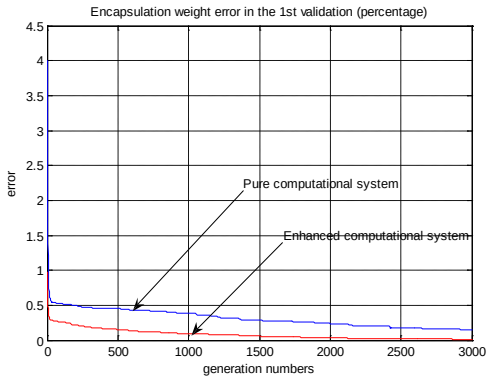

Figure 6 Search results of Validation test 1 for encapsulation weight

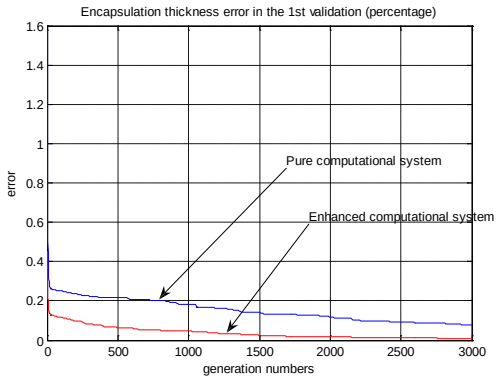

Figure 7 Search results of Validation test 1 for encapsulation thickness

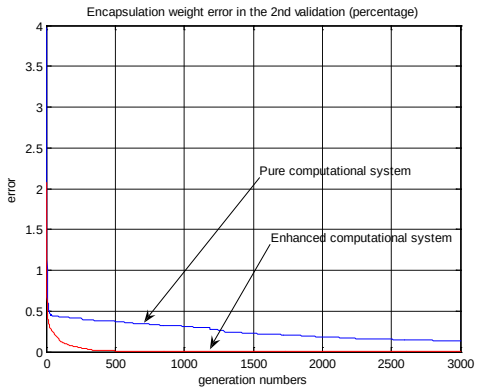

Figure 8 Search results of Validation test 2 for encapsulation weight

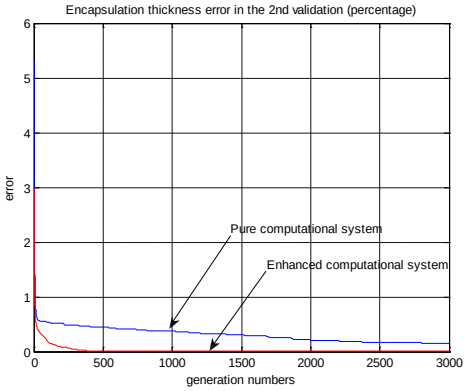

Figure 9 Search results of Validation test 2 for encapsulation thickness

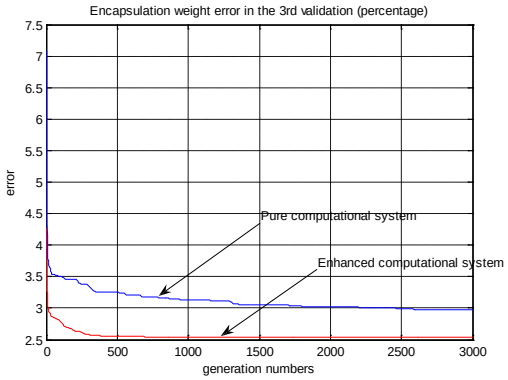

Figure 10 Search results of Validation test 3 for encapsulation weight

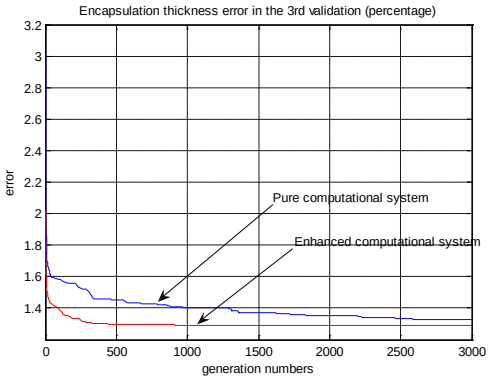

Figure 11 Search results of Validation test 3 for encapsulation thickness

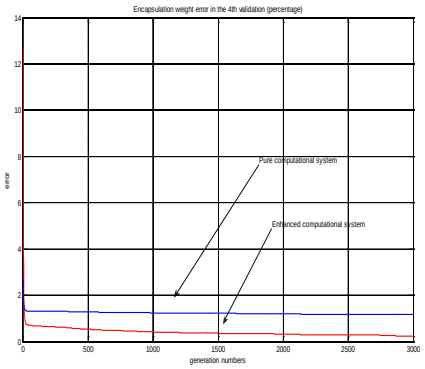

Figure 12 Search results of Validation test 4 for encapsulation weight

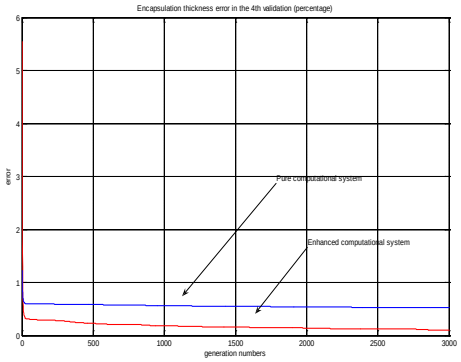

Figure 13 Search results of Validation test 4 for encapsulation thickness

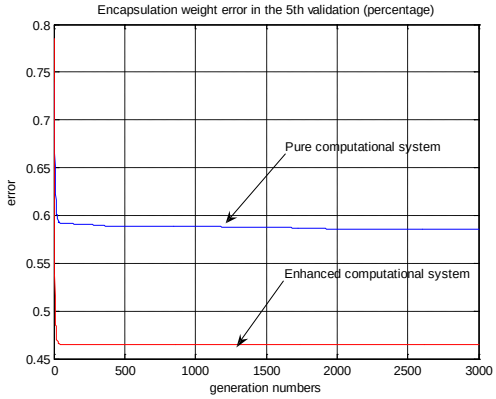

Figure 14 Search results of Validation test 5 for encapsulation weight

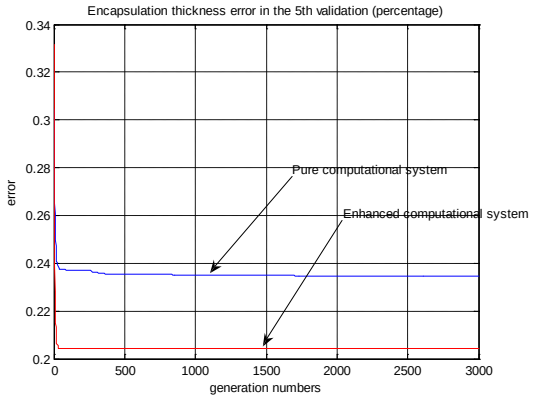

Figure 15 Search results of Validation test 5 for encapsulation thickness

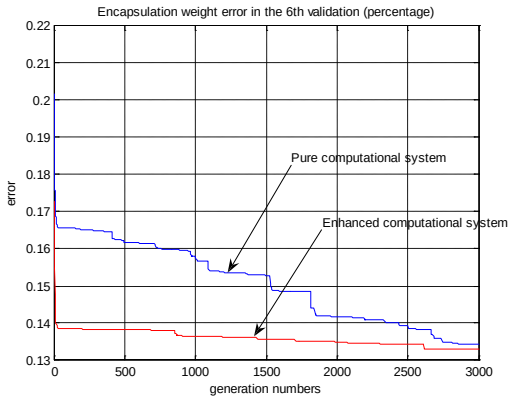

Figure 16 Search results of Validation test 6 for encapsulation weight

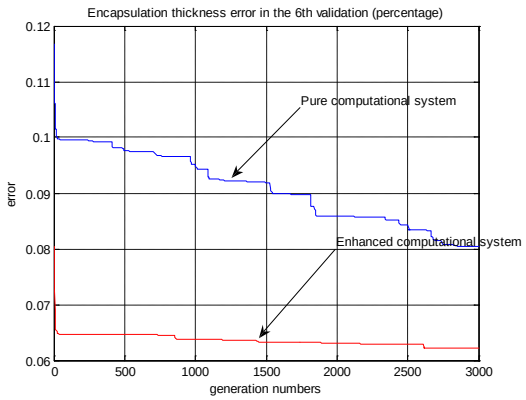

Figure 17 Search results of Validation test 6 for encapsulation thickness

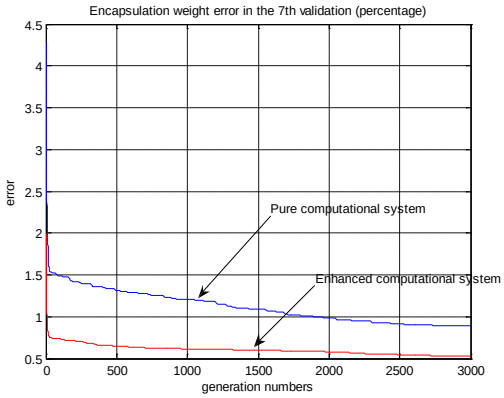

Figure 18 Search results of Validation test 7 for encapsulation weight

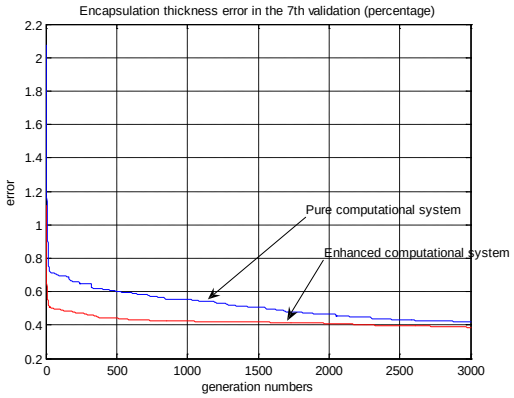

Figure 19 Search results of Validation test 7 for encapsulation thickness

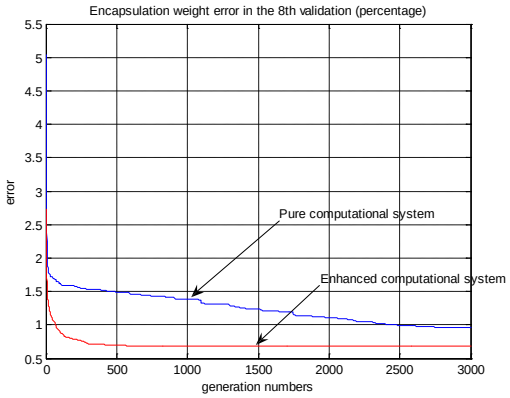

Figure 20 Search results of Validation test 8 for encapsulation weight

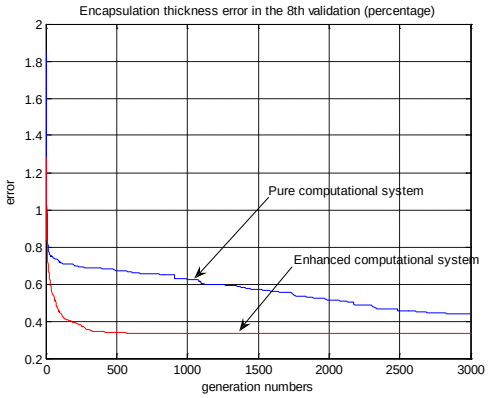

Figure 21 Search results of Validation test 8 for encapsulation thickness

Table 1 Training data sets for rule (4)

Data sets	y	z	x_1	x_2	x_3	**Class**
1-st	70.1	0.53	0.8	100	200	FN
2-nd	64.3	0.51	1.2	400	350	FP
3-rd	66.9	0.57	1.8	350	300	TP
4-th	65.5	0.61	1.6	40	220	TN

Table 2 Eight sets of required encapsulation weights and thickness

	Weight Y	Thickness z
1	72.3	0.58
2	43.2	0.48
3	87.4	0.67
4	37.2	0.46
5	75.1	0.62
6	59.3	0.57
7	62.4	0.53
8	53.1	0.53

Table 3 Means of the relative errors

	Mean errors of encapsulation weight		**Mean errors of encapsulation thickness**	
Validation	**Pure computational system (%)**	**Enhanced computational system (%)**	**Pure computational system (%)**	**Enhanced computational system (%)**
1	0.1847	0.0289	0.2377	0.0534
2	0.1796	0.0132	0.0886	0.0147
3	2.9548	2.5588	1.3575	1.2588
4	1.2272	0.3691	0.5150	0.1594
5	0.5632	0.4533	0.2360	0.2031
6	0.1345	0.1338	0.084	0.0064
7	0.8581	0.5141	0.4547	0.3945
8	0.9710	0.7174	0.4358	0.3527

Table 4 Variances of the relative errors

	Variances of errors of encapsulation weight		**Variances of errors of encapsulation thickness**	
Validation	**Pure computational system (%)**	**Enhanced computational system (%)**	**Pure computational system (%)**	**Enhanced computational system (%)**
1	0.6589×10^{-5}	0.0000×10^{-5}	0.1246×10^{-5}	0.0000×10^{-5}
2	0.7143×10^{-5}	0.0000×10^{-5}	0.1351×10^{-5}	0.0000×10^{-5}
3	0.3722×10^{-4}	0.0010×10^{-4}	0.3171×10^{-4}	0.0002×10^{-4}
4	0.9816×10^{-5}	0.0016×10^{-5}	0.3736×10^{-5}	0.0002×10^{-5}
5	0.2512×10^{-3}	0.0004×10^{-3}	0.2565×10^{-3}	0.0001×10^{-3}
6	0.8252×10^{-5}	0.0009×10^{-5}	0.3451×10^{-4}	0.0002×10^{-4}
7	0.1001×10^{-4}	0.0396×10^{-4}	0.0007×10^{-4}	0.0001×10^{-4}
8	0.2136×10^{-4}	0.1283×10^{-4}	0.0001×10^{-4}	0.0000×10^{-4}

Table 5 *t*-values between pure computational system and enhanced computational system for the relative errors of encapsulation weight and encapsulation thickness

Validation	**T-values of encapsulation weight between pure and enhanced computational systems**	**T-values of encapsulation thickness between pure and enhanced computational systems**
1	4.2918×10^{2}	1.1674×10^{3}
2	4.4024×10^{2}	4.4957×10^{2}
3	4.5836×10^{2}	1.2390×10^{2}
4	1.9351×10^{3}	1.3006×10^{3}
5	48.9922	14.5229
6	5.4459	93.3788
7	6.5079×10^{2}	1.5050×10^{3}
8	3.0667×10^{2}	5.8761×10^{3}