Genetic Simulated Annealing For Null Values Estimating in Generating Weighted Fuzzy Rules From Relational Database Systems

Irfan Subakti
Department of Informatics, Faculty of Information Technology
Institute Technology of Sepuluh Nopember Surabaya (ITS)
Kampus ITS, Keputih, Sukolilo, Surabaya, Indonesia
yifana@gmail.com

Abstract

Several methods are proposed to estimate null values in relational database systems. Often the estimated accuracy of the existing methods is not good enough. In this paper, we present an improving method to generate weighted fuzzy rules from relational database systems for estimating null values using Genetic Simulated Annealing (GSA), where the attributes appearing in the antecedent part of generated fuzzy rules have different weights.

After a predefined number of evolutions of the GSA, the best chromosome contains the optimal weights of the attributes, and they can be translated into a set of rules to be used for estimating null values. The improving method can get a higher average estimated accuracy rate than the existing method (i.e., using Genetic Algorithms). The only constraint of GSA is time. It takes longer time comparing with Genetic Algorithms (GA). We also modified the Equation for the better result comparing with the existing method.

Keywords: Fuzzy sets, Genetic Simulated Annealing, null values, weighted fuzzy rules

1. Introduction

Many methods have been proposed to generate fuzzy rules from training instances [2], [5], [3], [8], [11], [12] based on the fuzzy set theory [13]. In [6] Chen and Huang have presented a method for generating fuzzy rules from relational database systems for estimating null values. However, the average estimated accuracy rate of the method is challenged to be improved.

In this paper, in order to improve the method presented in [6] using Genetic Simulated Annealing (GSA) [7], where the attributes appearing in the antecedent parts of the generated fuzzy rules have different weights. The proposed improving method uses GSA to adjust the weights of the attributes for estimating null values in relational database systems. It can get a higher average estimated accuracy than the existing methods in [6]

The rest of this paper is organized as follows. In Section 2, we briefly review the basic concepts of fuzzy sets from [13]. In Section 3, we briefly review a method for tuning the weights of the attributes appearing in the antecedent parts of the generated fuzzy rules using GA for estimating null values in relational database systems [6]. In Section 4, we describe our improving method and the evaluation comparing with [6]. Finally we conclude our research in Section 5.

2. Basic Concepts of Fuzzy Sets

Zadeh proposed the theory of fuzzy sets in 1965 [13]. In a fuzzy set, each element in the set is associated with a membership value between 0 and 1 described by a membership function to indicate the grade of membership of the element in the fuzzy set. There are two types of membership functions to represent fuzzy sets. One is the discrete type membership function, and the other is the continuous type membership function. Let U be the universe of discourse, $U = \{u_1, u_2, \ldots, u_n\}$. A fuzzy subset A of the universe of discourse U can be represented as follows:

$$A = \mu_A(u_1)/u_1+\mu_A(u_2)/u_2+ \ldots +\mu_A(u_n)/u_n \qquad (1)$$

where μ_A is the membership function of the fuzzy subset A, μ_A: U → [0,1], and $\mu_A(u_i)$ indicates the grade of membership of u_i in the fuzzy subset A. If the universe of discourse U is a continuous set, then the fuzzy subset A can be represented as follows:

$$A = \int_U \mu_A(u)/u, \quad u \in U \qquad (2)$$

A linguistic term can be represented by a fuzzy set represented by a membership function. In this paper, the membership functions of the linguistic terms "L," "SL," "M." "SH," and "H" of the attributes "Salary" and "Experience" in relational database system are adopted from [3] as shown in Fig. 1 and Fig. 2, respectively, where "L" denotes "Low," "SL" denotes "Somewhat Low," "M" denotes "Medium", "SH" denotes "Somewhat High," and "H" denotes "High". Assume that there is an employee whose salary is 48,000 dollars per month,

and assume that his working experience is 4 years, then according to Fig. 1, we can see that the degree of membership that his salary (i.e., 48,000 dollars) belonging to the linguistic terms "Medium" (M) and "Somewhat High" (SH) are 0.7 and 0.3, respectively. According to Fig. 2, we can see that the degrees of membership that his experience (i.e., 4 years) belonging to the linguistic terms of "Somewhat Low" (SL) and "Medium" (M) are 0.5 and 0.5, respectively. Let x be a real value, and let X and Y be two linguistic terms. Assume that the degrees of membership of x belonging to X and Y are a and b, respectively, where $a \in [0,1]$ and $b \in [0,1]$. Based on [3] and [12], if $a \geq b$, then the real employee's salary (i.e., 48,000 dollars) is fuzzified into M/0.7, and his experience (i.e., 4 years) is fuzzified into SL/0.5. If p is a nonnumeric datum, where $p \in$ {Bachelor, Master, PhD}, then according to [3], p is fuzzified into p/1.0.

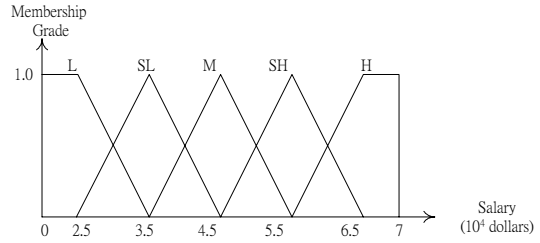

Figure 1. Membership functions of the linguistic terms of the attribute "Salary"

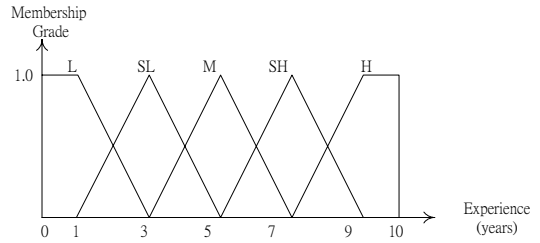

Figure 2. Membership functions of the linguistic terms of the attribute "Experience"

3. Tuning the Weight of the Attributes Using GAs [6]

3.1 Format of a Chromosome

Let's consider a relation of a relational database shown in Table 1. Based on Figs. 1 and 2, the values of attributes "Degree" and "Experience" shown in Table 1 can be fuzzified into Table 2. First, we define the format of a chromosome as shown in Fig. 4, where the value of each gene in a chromosome is a real value between zero and one, and the 13^{th} gene labeled "B-L" denotes the fuzzified values of the attributes "Degree" and "Experience" are "Bachelor" (B) and "Low" (L), respectively.

Table 1. Relation in a relational database

EMP-ID	Degree	Experience	Salary
S1	Ph.D.	7.2	63,000
S2	Master	2.0	37,000
S3	Bachelor	7.0	40,000
S4	Ph.D.	1.2	47,000
S5	Master	7.5	53,000
S6	Bachelor	1.5	26,000
S7	Bachelor	2.3	29,000
S8	Ph.D.	2.0	50,000
S9	Ph.D.	3.8	54,000
S10	Bachelor	3.5	35,000
S11	Master	3.5	40,000
S12	Master	3.6	41,000
S13	Master	10.0	68,000
S14	Ph.D.	5.0	57,000
S15	Bachelor	5.0	36,000
S16	Master	6.2	50,000
S17	Bachelor	0.5	23,000
S18	Master	7.2	55,000
S19	Master	6.5	51,000
S20	Ph.D.	7.8	65,000
S21	Master	8.1	64,000
S22	Ph.D.	8.5	70,000

Table 2. Fuzzified results of the value of the attributes "Degree" and "Experience"

EMP-ID	Degree	Experience	Salary
S1	Ph.D./1.0	SH/0.9	63,000
S2	Master/1.0	L/0.5	37,000
S3	Bachelor/1.0	SH/1.0	40,000
S4	Ph.D./1.0	L/0.9	47,000
S5	Master/1.0	SH/0.75	53,000
S6	Bachelor/1.0	L/0.75	26,000
S7	Bachelor/1.0	SL/0.65	29,000
S8	Ph.D./1.0	L/0.5	50,000
S9	Ph.D./1.0	SL/0.6	54,000
S10	Bachelor/1.0	SL/0.75	35,000
S11	Master/1.0	SL/0.75	40,000
S12	Master/1.0	SL/0.7	41,000
S13	Master/1.0	H/1.0	68,000
S14	Ph.D./1.0	M/1.0	57,000
S15	Bachelor/1.0	M/1.0	36,000
S16	Master/1.0	SH/0.6	50,000
S17	Bachelor/1.0	L/1.0	23,000
S18	Master/1.0	SH/0.9	55,000
S19	Master/1.0	SH/0.75	51,000
S20	Ph.D./1.0	SH/0.6	65,000
S21	Master/1.0	H/0.55	64,000
S22	Ph.D./1.0	H/0.75	70,000

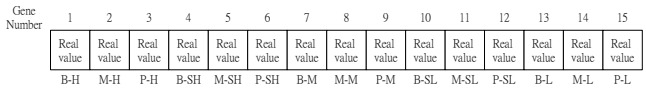

Figure 4. Format of a chromosome

From Fig. 4, we can see that each chromosome represents a combination of the weights of attributes, and it is a string of the weights of the attributes which will be used to estimate null values in relational databases systems. A population contains a set of chromosomes, and we can arbitrary set the number of chromosomes in a population. A chromosome consists of 15 genes. Because the total weights of attributes must be equal to one, the weight of attribute "Experience" must equal to one minus the weight of attribute "Degree". For example, let's see Fig. 5.

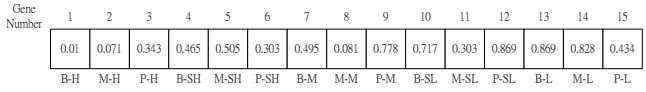

Figure 5. Example of a chromosome

Therefore, the content of the chromosome shown in Fig. 5 can be translated into the following 15 rules.

Rule 1: **IF** Degree=Bachelor **AND** Experience=High, **THEN** the Weight of Degree=0.010 **AND** the Weight of Experience=0.99.

Rule 2: **IF** Degree=Master **AND** Experience=High, **THEN** the Weight of Degree=0.071 **AND** the Weight of Experience=0.929.

…

Rule 15: **IF** Degree=PhD **AND** Experience=Low, **THEN** the Weight of Degree=0.434 **AND** the Weight of Experience=0.566.

3.2 Calculation of the Fitness Degree

Assume that there are n tuples T_1, T_2, …, T_n in a relation R of a relational database system, where the value of the attribute "Salary" of tuple T_i is denoted as "T_i.Salary." Let "ET_i.Salary" denotes the estimated value of T_i.Salary. In order to derive the value of ET_i.Salary, it must find a tuple T_j which is closest to the tuple T_i regarding the values of the attributes "Degree" and "Experience."

Table 3. Degree of similarity between the values of the attribute "Degree"

	Bachelor	Master	PhD
Bachelor	1	0.6	0.4
Master	0.6	1	0.6
PhD	0/4	0.6	1

Based on a fuzzy similarity matrix, it can obtain the degree of similarity between two nonnumeric values. For example, from Table 3, we can see that the degree of similarity between the degrees "Bachelor" and "PhD" is 0.4. From [4], we can see that the ranks of the terms "Bachelor," "Master," and "PhD" are 1, 2 and 3, respectively. That is:

$$\text{Rank(Bachelor)} = 1$$
$$\text{Rank(Master)} = 1$$
$$\text{Rank(PhD)} = 1$$

Let X be a nonnumeric attribute. Based on the value T_i.X of the attribute X of tuple T_i and the value T_j.X of the attribute X of tuple T_j, where $i \neq j$, the degree of closeness Closeness(T_i, T_j) between tuples T_i and T_j can be calculated by (3) or (4), where Weight(T_j.Degree) and Weight(T_j.Experience) denote the weights of the attributes "Degree" and "Experience," respectively, obtained from the fuzzified values of the attributes "Degree" and "Experience" of tuple T_j, derived from a chromosome.

If Rank(T_i.X) ≥ Rank(T_j.X) then

$$\text{Closeness}(T_i, T_j) = \text{Similarity}(T_i.X, T_j.X) \times \text{Weight}(T_j.\text{Degree}) + \frac{T_i.Experience}{T_j.Experience} \times \text{Weight}(T_j.\text{Experience}) \quad (3)$$

If Rank(T_i.X) < Rank(T_j.X) then

$$\text{Closeness}(T_i, T_j) = 1/\text{Similarity}(T_i.X, T_j.X) \times \text{Weight}(T_j.\text{Degree}) + \frac{T_i.Experience}{T_j.Experience} \times \text{Weight}(T_j.\text{Experience}) \quad (4)$$

where Similarity(T_i.X, T_j.X) denotes the degree of similarity between T_i.X and T_j.X, and its value is obtained from a fuzzy similarity matrix of the linguistic terms of the attribute X defined by a domain expert.

Let T_i, T_j and T_k be three tuples in a relational database. Assume that the degree of closeness between tuple T_i and T_j is denoted as Closeness(T_i, T_j), and the degree of closeness between tuples T_i and T_k is denoted as Closeness(T_i, T_k). Let $x = |\text{Closeness}(T_i, T_j)-1|$ and let $y = |\text{Closeness}(T_i, T_k)-1|$. If $x < y$, then tuple T_j is more close to tuple T_i and all the other tuples in the relational database have been calculated, the tuple whose closeness degree with respect to tuple T_i is closest to 1.0 is regarded as closest to tuple T_i, where $1 \le i \le n$.

After calculating the degrees of closeness of the other tuples in the database with respect to tuple T_i, the system will pick a tuple which is closest to tuple T_i. Assume that tuple T_j is closest to tuple T_i, then we can calculate the estimated value "ET_i.Salary" of the attribute "Salary" of tuple T_i as follows:

$$ET_i.\text{Salary} = T_i.\text{Salary} \times \text{Closeness}(T_i, T_j) \quad (5)$$

where T_i.Salary denotes the value of the attribute "Salary" of tuple T_j.

In the same way, it can calculate the estimated values of the attribute "Salary" of all the tuples in a relational database. After we get the estimated values of the attribute "Salary" of all tuples in a relational database, it can calculate the estimated error of each tuple by (6), where $Error_i$ denotes the estimated error between the estimated value ET_i.Salary of the attribute "Salary" of tuple T_i and the actual value T_i.Salary of the attribute "Salary" of tuple T_i

$$\text{Error}_i = \frac{ET_i.Salary - T_i.Salary}{T_i.Salary} \quad (6)$$

Let Avg_Error denotes the average estimated error of the tuples based on the combination of weights of the attributes derived from the chromosome, where

$$\text{Avg\_Error} = \frac{\sum_{i=1}^{n} Error_i}{n} \quad (7)$$

Then, it can obtain the fitness degree of this chromosome as follows:

$$\text{Fitness Degree} = 1 - \text{Avg\_Error} \quad (8)$$

4. Estimating Null Values in Relational Database Systems Using GSA

GSA proposed by Koakutsu *et al.* [7]. It generates the seeds of Simulated Annealing (SA) sequentially, that is the seeds of a SA local search depends of the best-so-far solutions of all previous SA local searches. This sequentially approach seems to generate better child solutions.

GSA uses fewer crossover operations since it only uses crossover operations when the SA local search reaches a flat surface and it is time to jump in the solution space.

GSA starts with a population $X = \{x_1, \ldots, X_{Np}\}$ and repeatedly applies 3 operations: SA-based [10] local search, GA-based [1] [9] crossover operation, and population update. SA-based local search produces a candidate solution x' by changing a small fraction of the state of x. The candidate solution is accepted as the new solution with probability $\min\{1, e^{-\Delta f/T}\}$. GSA preserves the local best-so-far solution x^*_L during the SA-based local search. When the search reaches a flat surface or the system is frozen, GSA produces a large jump in the solution space by using GA-based crossover. GSA picks up a pair of parent solutions x_j and x_k at random from the population X such that $f(x_j) \neq f(x_k)$, applies crossover operator, and then replace the worst solution x_i by the new solution produced by the crossover operator. At the end of each SA-based local search, GSA updates the population by replacing the current solution x_i by the local best-so-far solution x^*_L. GSA terminates when the CPU time reaches given limit, and reports the global best-so-far solution x^*_G.

Pseudo-code of GSA [7] is as follows:

```
GSA_algorithm(Np, Na, T0, α)
{
  X ← {x1, ..., XNp}; /* initialize population */
  x*L ← the best solution among X; /* initialize local best-so-far */
  x*G ← x*L /* initialize global best-so-far */
  while (not reach CPU time limit) {
    T ← T0; /* initialize temperature */
    /* jump */
    select the worst solution xi from X;
    select two solutions xj, xk from X such that f(xj) ≠ f(xk);
    xi ← Crossover(xj, xk);
    /* SA-based local search */
    while (not frozen or not meet stopping criterion) {
      for (loop = 1; loop ≤ Na; loop++) {
        x' --> Mutate(xi);
               Δf ← f(x') – f(xi);
               r ← random number between 0 and 1
               if (Δf < 0 or r < exp(-Δf/T))
                 xi ← x';
               if (f(xi) < f(x*L))
                 x*L --> xi; /* update local best-so-far */
             }
             T ← T * α /* lower temperature */
    }
    if (f(x*L) < f(x*G))
      x*G ← x*L; /* update global best-so-far */
    /* update population */
    xi ← x*L;
    f(x*L) ← +∞; /* reset current local best-so-far */
  }
  return x*G;
}
```

We implemented that pseudo-code by the code algorithm depicted at Figs. 6-9 below.

```
Procedure EvaluationAndBestSelection
    {find the best solution among population. Also it initializes
LocalBestChromosomeSoFar and GlobalBestChromosomeSoFar:
    X ← {x1, ..., XNp}; {initialize population}
    x*L ← the best solution among X; {initialize local best-so-far}
    x*G ← x*L {initialize global best-so-far}
    FitnessDegreeEval ← FitnessDegree from global best-so-far
    }
    for i:= 1 to number-of-generations do begin
      T ← T0;
      EvaluationAndWorstSelection; {select the worst solution xi from X}
      CrossOver; {select two solutions xj, xk from X such that f(xj) ≠ f(xk):
       xi ← Crossover(xj, xk);
      }
      Mutation; {update local best-so-far if value is better
       repeat
         for i:= 0 to number-of-mutation do begin
           f(xi) ← Get Fitness Degree from chromosome before mutation
           x' ← Mutate(xi)
           f(x') ← Get Fitness Degree from chromosome after mutation
           Δf ← f(xi) - f(x')
           r ← random number between 0 and 1
           ft ← f(x')
           if (Δf >= 0) or (r >= exp(-Δf/T)) then begin
             xi ← x';
             ft ← f(xi);
           end;
           if (ft >= FitnessDegreeEval) then begin
             x*L ← xi; {update local best-so-far}
             FitnessDegreeEval ← ft
             FDLocalBestSoFar ← ft {Get local best Fitness Degree}
           end
         end
         T ← T * α; {lower temperature}
       until T <= FrozenValue;
      }
      CountCloseness(x*L); {get FD from LocalBestChromosomeSoFar}
      AvgError:= AvgError / NumData;
      FDLocalBestSoFar:= 1 - AvgError;
      CountCloseness(x*G); {get FD from GlobalBestChromosomeSoFar}
      AvgError:= AvgError / NumData;
      FDGlobalBestSoFar:= 1 - AvgError;
      if FDLocalBestSoFar >= FDGlobalBestSoFar then begin
       x*G ← x*L; {update global best-so-far}
       FitnessDegreeEval:= FDGlobalBestSoFar;
      end;
      xi ← x*L; {update population}
    end;
```

Figure 6. GSA code algorithm

```
Procedure CountClosenessValue
    AvgError:= 0.0;
    for i:= 0 to NumData - 1 do begin  {base on all data available}
     BestClosenessEval:= MaxInt;
     IdxClosestCloseness:= i;
     for j:= 0 to NumData - 1 do
      if i <> j then begin
       if Rank(Ti.X) ≥ Rank(Tj.X) then begin
        ClosenessE(Ti,Tj)= Similarity(Ti.X,Tj.X) × Weight(Tj.Degree) +
          (Ti.Experience/Tj.Experience) × Weight(Tj.Experience);
       end
       else begin {If Rank(Ti.X) < Rank(Tj.X)}
        ClosenessE:= 1/Similarity(Ti.X,Tj.X) × Weight(Tj.Degree) +
          (Ti.Experience/Tj.Experience) × Weight(Tj.Experience);
       end;
       {find a tuples which is closest to 1.0 as a}
       {closest tuple to tuple Ti}
       ClosestCloseness:= Abs(1 - ClosenessE);
       if ClosestCloseness <= BestClosenessEval then begin
        BestClosenessEval:= ClosestCloseness;
        IdxClosestCloseness:= j;
       end;
      end;
      {Then we find Estimated Salary and Error for every record}
      {if this record was null value, so we must find}
      {another record that closest to 1}
     if IsNullValue(i) and IsNullValue(IdxClosestCloseness) then begin
      PreferIdx:= GetPreferIdx;
      ETi.Salary:= Ti. Salary × GetClosenessValue(PreferIdx);
```

```
  if T_prefer-index.Salary <> 0 then
    Error_i:= (ET_i.Salary – T_prefer-index.Salary)/ T_prefer-index.Salary
 end
 else begin
  ET_i.Salary:= T_i. Salary × GetClosenessValue(IdxClosestCloseness);
  if T_i.Salary <> 0 then
    Error_i:= (ET_i.Salary – T_iSalary)/ T_iSalary
 end;
 AvgError:= AvgError + Abs(Error_i);
end;
```

Figure 7. Procedure CountCloseness code algorithm

```
function GetClosenessValue(Idx)
  Result ← find value in ClosenessE which have the same index with Idx
```

Figure 8. Function GetClosenessValue code algorithm

```
function GetPreferIdx
  Result ← find value in ClosenessE that closest to 1, and it's not null
value
```

Figure 9. Function GetPreferIdx code algorithm

We revise the Equation (8), since we obtained that Equation (8) reduce precision of fitness value of a chromosome. It makes convergence slower, and the result becomes less precise. For example, once upon a time we have Avg_Error = - 0.1, so we get Fitness Degree = 1 – (-0.1) = 1.1. Another time if we have Avg_Error = 0.1, then we get Fitness Degree = 1 – 0.1 = 0.9. Both of Avg_Error (i.e., -0.1 and 0.1) are the same (i.e., deviation value) but obviously we get different result (i.e., 1.1 and 0.9, respectively). Within running it will slower to reach the convergence. So we proposed a bit correction as:

$$\text{Fitness Degree} = 1 - \text{Absolute (Avg\_Error)} \qquad (9)$$

As we predict before, after we make a bit corrections above and implemented with GSA Program, we get the result more precise, as experiences in the following showed.

Using the same relation in a relational database containing a null value [6] as shown in Table 4, we performed some experiences. We run this program for different parameters, each for 10 times, for 2 types of experiment. Table 5 described the parameters. Also the Size of Population and Number of Generations are varied for each running program.

Table 4. Relation in a relational database containing a null value

EMP-ID	Degree	Experience	Salary
S1	Ph.D.	7.2	63,000
S2	Master	2.0	37,000
S3	Bachelor	7.0	40,000
S4	Ph.D.	1.2	47,000
S5	Master	7.5	53,000
S6	Bachelor	1.5	26,000
S7	Bachelor	2.3	29,000
S8	Ph.D.	2.0	50,000
S9	Ph.D.	3.8	54,000
S10	Bachelor	3.5	35,000
S11	Master	3.5	40,000
S12	Master	3.6	41,000
S13	Master	10.0	68,000
S14	Ph.D.	5.0	57,000
S15	Bachelor	5.0	36,000
S16	Master	6.2	50,000
S17	Bachelor	0.5	23,000
S18	Master	7.2	55,000
S19	Master	6.5	51,000
S20	Ph.D.	7.8	65,000
S21	Master	8.1	64,000
S22	Ph.D.	8.5	null

Table 5. Parameters of experiments

Experiment	Mutation Rate	Initial Temperature	Alpha	Frozen Value
Type 1	0.01	100	0.7	0.00001
Type 2	0.1	100	0.7	0.00001

The result of one of experiment is depicted in Table 6, and the summaries from all experiments are depicted in Table 7.

Table 6. One result of the running program

Size of Population: 60. Number of Generations: 300
Mutation Rate: 0.01 Initial Temperature: 100
Alpha: 0.7 Frozen Value: 0.00001

#	Avg. Estimated Error	Hour	Minute	Second	Milisecond	Total Time
1	0.01365808	0	20	28	453	0h:20m:28s:453ms
2	0.01365808	0	20	32	328	0h:20m:32s:328ms
3	0.004348226	0	20	29	500	0h:20m:29s:500ms
4	0.004348226	0	20	30	125	0h:20m:30s:125ms
5	0.004348226	0	20	27	78	0h:20m:27s:78ms
6	0.004348226	0	20	23	282	0h:20m:23s:282ms
7	0.004348226	0	20	27	984	0h:20m:27s:984ms
8	0.004348226	0	20	30	328	0h:20m:30s:328ms
9	0.004348226	0	20	31	578	0h:20m:31s:578ms
10	0.004348226	0	20	30	16	0h:20m:30s:16ms
Min	0.004348226	0	20	23	16	
Average	0.006210197	0	20	28.7	367.2	
Max	0.01365808	0	20	32	984	

Table 7. Summaries of all experiments
(a)

Avg. Estimation Error	Size Of Population	Number of Generations	Mutation Rate (%)	Initial Temperature	Alpha	Frozen Value
0.002967128	60	300	10	100	0.7	0.00001
0.005444614	40	150	10	100	0.7	0.00001
0.005466245	30	100	10	100	0.7	0.00001
0.006210197	60	300	1	100	0.7	0.00001
0.006618262	50	200	10	100	0.7	0.00001
0.006621554	40	150	1	100	0.7	0.00001
0.007580152	50	200	1	100	0.7	0.00001
0.008184308	30	100	1	100	0.7	0.00001

(b)

Min. Estimation Error	Size Of Population	Number of Generations	Mutation Rate (%)	Initial Temperature	Alpha	Frozen Value
0.002890637	60	300	10	100	0.7	0.00001
0.003153504	40	150	1	100	0.7	0.00001
0.003194167	30	100	10	100	0.7	0.00001
0.003410507	40	150	10	100	0.7	0.00001
0.003821698	50	200	10	100	0.7	0.00001
0.004348226	60	300	1	100	0.7	0.00001
0.006138154	30	100	1	100	0.7	0.00001
0.006334942	50	200	1	100	0.7	0.00001

(c)

Max. Estimation Error	Size Of Population	Number of Generations	Mutation Rate (%)	Initial Temperature	Alpha	Frozen Value
0.003090779	60	300	10	100	0.7	0.00001
0.007768472	40	150	10	100	0.7	0.00001
0.00905076	30	100	10	100	0.7	0.00001
0.009636727	50	200	1	100	0.7	0.00001
0.009740355	50	200	10	100	0.7	0.00001
0.011162399	30	100	1	100	0.7	0.00001
0.01365808	60	300	1	100	0.7	0.00001
0.018120004	40	150	1	100	0.7	0.00001

4.1 Chen's [6] Result

As a comparison, we toke the results from Chen *et al.* [6] as in the following.

Best chromosome:

0.010 0.071 0.343 0.465 0.505 0.303
0.495 0.081 0.778 0.717 0.303 0.869
0.869 0.828 0.434

Below is the result from with size of population: 60; number of generations: 300; Cross Over rate: 1.0; and Mutation rate: 0.2

EMP-ID	Degree	Experience	Salary	Salary (Estimated)	Estimated Error
S1	Ph.D.	7.2	63,000	61,515.00	-0.024
S2	Master	2.0	37,000	36,967.44	-0.001
S3	Bachelor	7.0	40,000	40,634.14	0.016
S4	Ph.D.	1.2	47,000	46,873.66	-0.003
S5	Master	7.5	53,000	56,134.37	0.059
S6	Bachelor	1.5	26,000	26,146.40	0.006
S7	Bachelor	2.3	29,000	27,822.08	-0.041
S8	Ph.D.	2.0	50,000	50,067.20	0.001
S9	Ph.D.	3.8	54,000	53,958.94	-0.001
S10	Bachelor	3.5	35,000	35,152.00	0.004
S11	Master	3.5	40,000	40,206.19	0.005
S12	Master	3.6	41,000	40,796.57	-0.005
S13	Master	10.0	68,000	68,495.74	0.007
S14	Ph.D.	5.0	57,000	56,240.72	-0.013
S15	Bachelor	5.0	36,000	34,277.54	-0.048
S16	Master	6.2	50,000	49,834.85	-0.003
S17	Bachelor	0.5	23,000	23,722.40	0.031
S18	Master	7.2	55,000	51,950.6	-0.055
S19	Master	6.5	51,000	51,197.58	0.004
S20	Ph.D.	7.8	65,000	64,813.75	-0.003
S21	Master	8.1	64,000	60,853.28	-0.049
S22	Ph.D.	8.5	70,000	69,065.83	-0.013
Average Estimated Error					0.018

For another running it also gets the average estimated errors for different parameters of the GA.

Size of Population	Number of Generations	Crossover Rate	Mutation Rate	Average Estimated Error
30	100	1.0	0.1	0.036
40	150	1.0	0.1	0.032
50	200	1.0	0.2	0.027
60	300	1.0	0.2	0.018

4.2 This Improving Method's Result

Size of Population: 60
Number of Generations: 300
Mutation Rate (%): 10
Initial Temperature: 100
Alpha: 0.7
Frozen Value: 0.00001

Best Chromosome

Gene-1	Gene-2	Gene-3	Gene-4	Gene-5	Gene-6
0.719	0.995	0.989	0.485	0.095	0.896

Gene-7	Gene-8	Gene-9	Gene-10	Gene-11
0.277	0.416	0.085	0.997	0.183

Gene-12	Gene-13	Gene-14	Gene-15
0.583	0.350	0.652	0.241

EMP-ID	Degree	Experience	Salary	Salary (Estimated)	Estimated Error
1	Ph.D.	7.2	63,000	62,889.86	-0.00174820
2	Master	2	37,000	36,847.97	-0.00410900
3	Bachelor	7	40,000	40,128.33	0.00320820
4	Ph.D.	1.2	47,000	46,538.60	-0.00981700
5	Master	7.5	53,000	52,978.58	-0.00040420
6	Bachelor	1.5	26,000	25,970.00	-0.00115400
7	Bachelor	2.3	29,000	28,967.01	-0.00113750
8	Ph.D.	2	50,000	50,341.15	0.00682300
9	Ph.D.	3.8	54,000	53,836.28	-0.00303190
10	Bachelor	3.5	35,000	35,060.59	0.00173100
11	Master	3.5	40,000	39,876.06	-0.00309860

12	Master	3.6	41,000	40,875.72	-0.00303120
13	Master	10	68,000	68,087.03	0.00127980
14	Ph.D.	5	57,000	56,731.71	-0.00470680
15	Bachelor	5	36,000	36,051.19	0.00142190
16	Master	6.2	50,000	49,936.01	-0.00127980
17	Bachelor	0.5	23,000	22,940.28	-0.00259660
18	Master	7.2	55,000	54,966.66	-0.00060620
19	Master	6.5	51,000	50,945.03	-0.00107780
20	Ph.D.	7.8	65,000	64,938.81	-0.00094140
21	Master	8.1	64,000	63,933.65	-0.00103670
22	Ph.D.	8.5	70,000	70,654.72	0.00935310
Average Estimated Error					0.002890636946022
Time Elapsed					3h:15m:50s:968ms

We can see from the result that by using GSA it improves the result of the existing method (i.e., using Genetic Algorithms) proposed by Chen *et al.* [6].

We can't get the time elapsed by the GA method used in the [6], but during the implementation of our method we also try to use GA and observed the performance. Apparently, by using GSA the only constraint is time. GSA obviously takes longer time than GA while obtaining significant best value of chromosome through the decrement of T (temperature), since the decrement should be maintained bit by bit consistently until reach the frozen value. If the frozen value is too high, the result will decrease significantly, but if it's too low, it takes a significant time even though the result is quite higher. So, there's a trade-off for deciding the frozen value for GSA.

5. Conclusions

In this paper, we have presented an improving method to estimate null values in relational database systems using Genetic Simulated Annealing (GSA), where the attributes appearing in the antecedent parts of the generated fuzzy rules have different weights.

After a predefined number of evolutions of the GSA, the best chromosome contains optimal weights of the attributes, and they can be translated into a set of rules to be used for estimating null values. This improving method can get a higher average estimated accuracy rate than the existing method which used GA. The only constraint of GSA is time. It takes longer time comparing with GA.

We also modified Equation (8) to (9) to make the convergence process faster and the result of estimating null values in relational database systems using GSA becomes more precise.

6. References

[1] Brown, D., Huntley, C. and Spillane, A. , "A parallel genetic heuristic for the quadratic assignment problem," *Proc. 3rd Int. Conf. Genetic Algorithms*, 1989, pp.406-415.

[2] Burkhardt, D.G and Bonissone, P.P., "Automated fuzzy knowledge base generation and tuning," *Proc. 1992 IEEE Int. Conf. Fuzzy Systems*, San Diego, CA, 1992, pp. 179-419.

[3] Chen, S.M. and Yeh, M.S., "Generating fuzzy rules from relational database systems for estimating null values," *Cybern. Syst.*, vol. 28, no. 8, 1997, pp. 695-723.

[4] Chen, S.M. and Chen, H.H., "Estimating null values in the distributed relational database environment," *Cybern. Syst.*, vol. 31, no. 8, 2000, pp. 851-871.

[5] Chen, S.M., Lee, S.H. and Lee, C.H., "A new method for generating fuzzy rules from numerical data for handling classification problems," *Appl. Art. Intel.*, vol. 15, no. 7, 2001, pp. 645-664.

[6] Chen, S.M. and Huang, C.M., "Generating weighted fuzzy rules from relational database systems for estimating null values using genetic algorithms," *IEEE Trans.On Fuzzy Systems*, vol. 11, no. 4, 2003, pp. 495-506.

[7] Koakutsu, S., Kang, M. and Dai, W.W.-M, "Genetic simulated annealing and application to non-slicing floorplan design," *Proc. 5th ACM/SIGDA Physical Design Workshop*, (Virginia, USA), 1996, pp. 134-141.

[8] Lin, H.L. and Chen, S.M., "Generating weighted fuzzy rules from training data for handling fuzzy classification problems," *Proc. 2000 Int. Computer Symp.: Workshop Artificial Intelligence*, Chiayi, Taiwan, R.O.C., 2000, pp. 11-18.

[9] Lin, T.-T., Kao, C.-Y. and Hsu, C.-C., "Applying the genetic approach to simulated annealing in solving some NP-Hard problems," *IEEE Trans. System, Man, and Cybernetics*, vol. 23, no. 6, 1993, pp.1752-1767.

[10] Sigrag, D. and Weisser, P., "Toward a unified thermodynamic genetic operator," *Proc. 2nd Int. Conf. Genetic Algorithms*, 1987, pp.116-122.

[11] Tsukimoto, H., "Extracting rules from trained neural networks," *IEEE Trans. Neural Networks*, vol. 11, 2000, pp. 377-389.

[12] Wang, L.X. and Mendel, J.M., "Generating fuzzy rules by learning from examples," *IEEE Trans. Syst., Man, Cybern.*, vol. 22, 1992, pp. 1414-1427.

[13] Zadeh, L.A., "Fuzzy sets," *Inform. Control*, vol. 8, 1965, pp. 338-353.