Coevolutionary GA with schema extraction by machine learning techniques and its application to knapsack problems

Hisashi Handa
Okayama University

Tadashi Horiuchi
Osaka University

Osamu Katai
Kyoto University

Takeshi Kaneko
Kyoto University

Tadataka Konishi
Okayama University

Mitsuru Baba
Okayama University

Coevolutionary GA with Schema Extraction by Machine Learning Techniques and Its Application to Knapsack Problems

H. Handa+ **T. Horiuchi*** **O. Katai**** **T. Kaneko**** **T. Konishi+** **M. Baba+**

+Faculty of Engineering
Okayama University
Tsushima-naka 3-1-1, Okayama
700-8530, JAPAN
handa@sdc.it.okayama-u.ac.jp
+81-86-251-8250

*I.S.I.R.
Osaka University
Mihogaoka 8-1, Ibaraki, Osaka
567-0047, JAPAN
horiuchi@ar.sanken.osaka-u.ac.jp
+81-6-6879-8542

**Graduate School of Informatics
Kyoto University
Yoshida-Honmachi, Kyoto
606-8501, JAPAN
katai@i.kyoto-u.ac.jp
+81-75-753-5201

Abstract- In this paper, we will introduce a new Coevolutionary Genetic Algorithm with schema extraction by machine learning techniques. Our CGA consists of two GA populations: the first GA (H-GA) searches for the solutions in the given problems and the second GA (P-GA) searches for effective schemata of the H-GA. We aim to improve the search ability of our CGA by extracting more efficiently useful schemata from the H-GA population, and then incorporating those extracted schemata in natural manner into the P-GA. Several computational simulations on multidimensional knapsack problems confirm us the effectiveness of the proposed method.

1 Introduction

In this paper, we will introduce a new Coevolutionary Genetic Algorithm with schema extraction by machine learning techniques. Basic idea in this Coevolutionary Genetic Algorithm is to explore the effective genetic information in the population, i.e., schemata, and to exploit the genetic information in order to guide the population to better solutions [1][2]. Our Coevolutionary Genetic Algorithm consists of two GA populations which separately evolve and affect with each other; the first GA, called "H-GA" (Host GA), searches for the solutions in the given problem environment, and the second GA, called "P-GA" (Parasite GA), searches for the effective genetic information suitable for the H-GA, namely, good schemata. Thus, each individual in the P-GA consists of alleles in the H-GA and "don't care symbols" representing a schema in the H-GA. In order to realize the interaction between these two GA populations, we introduce two kinds of genetic operators, what we call, "superposition operator" and "transcription operator". Further details of our Coevolutionary Genetic Algorithm are described in the next section.

In this paper, we aim to improve the search ability of our Coevolutionary Genetic Algorithm by extracting more efficiently useful schemata from the H-GA populations. We will adopt a *Machine Learning* technique as a method to extract useful schemata from H-GA. The training data for the machine learning are of a large amount of genetic information and their fitness values which have been searched by the H-GA. In other words, we apply an *inductive learning* method for the past data which the H-GA has been searching, by regarding the individuals with highly enough fitness values as "positive instances" and the individuals with low enough fitness values as "negative instances". Then, we can derive a set of rules which characterizes (discriminates) the positive instances and negative instances. By focusing the rules whose consequent part is "positive class", their antecedent parts mean the pairs of an allele and its value which are important for characterizing the positive instances (individuals with highly enough fitness). Therefore, we can regard those pairs as parts of useful schemata. That is, we regard those antecedent parts in the rules derived by machine learning as useful schemata, and try to propagate those schemata information in the P-GA.

In general, the P-GA searches for the *useful schemata* in the H-GA. Here, the useful schemata in the H-GA may be defined as follows: (1) *undiscovered* useful schemata in the H-GA, or (2) useful schemata, i.e., those with high average fitness values and can be regarded as a part of building blocks. Our proposed method explicitly extracts the latter schemata in the above and make positive efforts to utilize those schemata through the genetic operations in P-GA.

0-7803-6657-3/01/$10.00

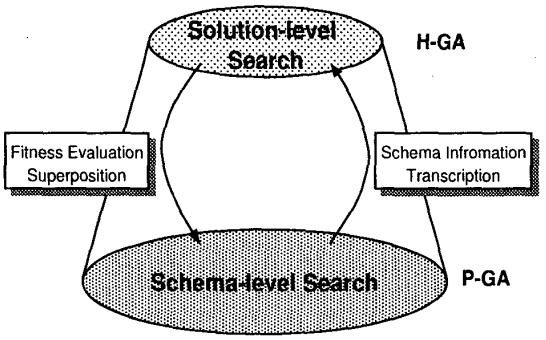

Figure 1: Process of Coevolutionary Genetic Algorithm

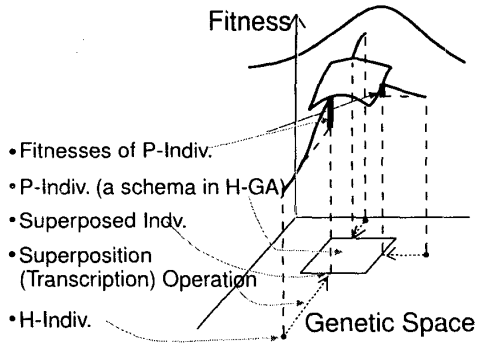

Figure 2: Fitness Evaluation of P-GA in H-GA's Genetic Space.

Related works are as follows: Schema-oriented search methods in evolutionary computation has been adopted in several problem solving methods such as Stochastic Schemata Exploiter [3] and Cultural Algorithms [4]. In the Stochastic Schemata Expoliter, schemata among the population is ranked. Then, highly ranked schemata are used for the next population by sampling individuals which belongs those schemata. In the Cultural Algorithm, usual GA model is associated with the beliefs space which is similar to the schema space and is used to promote directed evolution of individuals in the GA model. Our method is similar to Cultural Algorithm in the sense that both methods use additional mechanisms to promote the evolution of usual GA. The specificity of our approach is to use inductive learning method at every fixed interval instead of belief maintenace, and to use the mechanism of co-evolution.

2 Coevolutionary Genetic Algorithm

2.1 Framework

Our Coevolutionary Genetic Algorithm consist of two GA populations; the H-GA (Host GA), searches for the solutions in the given problem environment, and the P-GA (Parasite GA), searches for the effective genetic information involved in the H-GA, namely, good schemata. Thus, each individual in P-GA consists of alleles in H-GA and "don't care symbols" representing a schema in the H-GA. These two GA populations separately evolve in each genetic space at different abstraction levels and affect with each other by two genetic operators: "superposition" and "transcription". As depicted in Figure 1, these two genetic operators play the role to communicate (propagate) the genetic information between H-GA and P-GA. These operators are described in the following subsections.

2.2 Superposition Operator

The individuals of the P-GA called P-indiv.'s represent the schemata in the H-GA. Namely, each P-Indiv. consists of the alleles used in the H-GA and don't care symbols which represent a schema in the H-GA. The superposition operator copies the genetic information of a P-Indiv., except for the don't care symbols, onto one of H-Indiv.'s in order to calculate the fitness of the P-Indiv. Thus, the evolutionary process of this layered population can be regarded as co-evolution of individual at different levels of abstraction.

2.3 Fitness Evaluation of P-GA

In general, the fitness value of a schema is calculated as the average fitness value of the all individuals belonging to the schema. It is difficult, however, to calculate the fitness values of all individuals when the order of schema is small. Hence, the average value of schema is set to the average value of "sampled" individuals belonging to the schema. Furthermore, if the schema information discovered by P-GA is already discovered by H-GA, the H-GA will receive no effective information from this "discovery" by P-GA. Hence, we let P-GA to search for "undiscovered" useful schemata in H-GA, and the fitness evaluation of a P-Indiv. is given as follows: First, the fitness value F_j of a P-indiv., say, j is calculated in the following way[1]: The superposing (superposition) operation of

[1] In this paper, the fitness function of H-indiv. i and P-indiv. j are represented as f_i and F_j, respectively.

each P-indiv. onto H-indiv.'s is carried out n times. (1) First, n H-indiv.'s to be superposed by P-indiv. j are randomly selected. (2) These selected H-indiv.'s are denoted as $i_1, \ldots, i_n$, and the resultant superposed H-indiv.'s are denoted as $\tilde{i}_1, \ldots, \tilde{i}_n$. (3) Then, to calculate the fitness value of P-indiv. j, the effect of each of the superposition operations is evaluated the contribution of the superposition operation to each H-indiv. defined as follows:

$$F_j = \sum_{k=1}^{n} \max(0, f_{\tilde{i}_k} - f_{i_k}) \quad (k = 1, \ldots, n) .$$

Thick lines in Figure 2 denote the difference between the fitness values of the original H-indiv.'s and those of the superposed H-indiv.'s, that is, "positive contribution" of this superposition operation. If the difference is negative, then the contribution of this operation is regard to be 0.

2.4 Transcription Operator

The transcription operator serves as means for transmitting the effective genetic information searched by the P-GA to the H-GA. This operator propagates the genetic information in the P-GA to H-Indiv.'s by probabilistically replacing the original H-Indiv. i_k with superposed H-Indiv. $\tilde{i}_k$. The probability of applying the transcription operation is given as

$$P_{transcription} = \begin{cases} P_{parasite} \frac{\hat{f}_k}{f_{max} - f_{min}}, & \hat{f}_k > 0 \\ 0 & \text{otherwise,} \end{cases}$$

where f_{max} and f_{min} denote the maximum and the minimum fitness values in the H-GA, respectively, and hence $P_{parasite}$ is set to be constant such that $0 < P_{parasite} < 1$.

3 Schema Extraction by Machine Learning Techniques for Coevolutionary GA

In this paper, we propose to utilize a *Machine Learning* technique as a method to extract the useful schemata from the H-GA and to propagate those useful schemata in the P-GA, in order to improve the search ability of our Coevolutionary Genetic Algorithm. The central idea of this approach is to extract the useful schemata which characterize the high fitness individuals in the past populations which have been searched by the H-GA. In other words, machine learning can discover the useful schemata by regarding the past GA populations as the training data. Then, we aim to *guide* the direction of the evolution by adding the *bias* to the GA search through introducing the machine learning techniques for Coevolutionary Genetic Algorithm.

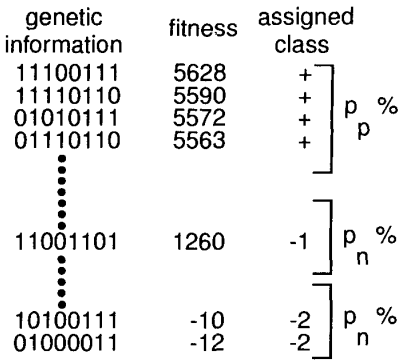

Figure 3: A depiction how to collect positive and negative instances

We can apply the inductive learning methods for the data which the H-GA has been searching, by regarding the individuals with highly enough fitness as "positive instances" and the individuals with low enough fitness as "negative instances". Then, we can derive a set of rules which can characterizes (discriminates) the positive instances and negative instances. By focusing the rules whose consequent part is "positive class", their antecedent parts mean the pairs of an allele and its value which are important for characterizing the positive instances (individuals with highly enough fitness)". Therefore, we can regard those pairs as candidates of useful schemata. That is, we regard those antecedent parts in the rules derived by machine learning as useful schemata, and try to propagate those schemata information in the P-GA.

Among the various inductive learning methods, we can utilize the learning methods which can learn a set of if-then rules. Several algorithms to learn a set of rules have been proposed so far in the area of machine learning, such as CN2 [5], RIPPER [6]. In this paper, we adopt C4.5 [7] which is a widely-used method for learning decision trees and classification rules.

Now, we concretely explain how to apply C4.5, one of the widely-used machine learning techniques, into Coevolutionary GA. The training data consists of the genotype of H-Indiv.'s which have been evaluated in GA

```
Rule 49:
        3 = 1
        6 = 0
        8 = 1
        ->  class +

Rule 3:
        2 = 1
        1 = 1
        9 = 1
        ->  class +

Rule 27:
        5 = 0
        4 = 0
        9 = 1
        ->  class -2
```

Figure 4: An example of output by command c4.5rules

search, i.e., all H-indiv.'s evaluated every generation and the genetic information (fitness information) generated by the superposition operation. Before applying C4.5, we need to divide those individuals into several group s (a single positive class and $n_c - 1$ negative classes in our case) according to their fitness values. As depicted in Figure 3, a positive class consists of the individuals with the highest p_p percent of the fitness values. The most negative class is also consists of the individuals with the lowest p_n percent of fitness values. The i-th negative class ($i = 1, 2, \cdots, n_c - 1$) is also made up of the individuals with the lower fitness values, one after another by p_n percent of the fitness values.

We use the commands "c4.5" and "c4.5rules" involved in the C4.5 programs to generate classification rules which indicate how genetic information, i.e., schema information, are used for characterizing the positive instances. Here, the command "c4.5" is a program for generating decision trees based on the information entropy, on the other hand, the generated decision tree are translated into a set of classification rules by using the command "c4.5rules". Classification rules acquired in the above manner are described in Figure 4. In this figure, "Rule 49" means 49th acquired rule by C4.5. And following three equations denote that "if the allele of the 3rd locus is equal to '1' and the one of the 6th locus is '1' and the one of 8th locus is '1'," and subsequent line explain that "then such pattern characterizes positive class". The acquired rules which explain the positive class, e.g., first and second rules in the figure, are incorporated in natural manner as the P-indiv. which is the same as schema information in the H-GA. For instance, the first rule in the figure is rewritten as the genetic information of P-indiv as follows.

$$**1**0*0*$$

4 Experiments and Results

4.1 Experiments

In this paper, we apply the proposed method to multidimensional knapsack problems which are regarded as a class of the combinatorial optimization problems. In general, multidimensional knapsack problems are formalized as follows,

$$\text{maximize} \quad \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i, \quad (i = 1, \ldots, m). \tag{2}$$

Here, c_j, a_{ij}, b_i are positive values and x_j is a binary value which takes 0 or 1.

In order to apply GAs to the above multidimensional knapsack problems, we adopt the following fitness function in the similar way used in Goldberg *et. al.* [8]:

$$F = O - C_{penalty} \times (\sum_{j=1}^{n} e_j)^2$$

where, O denotes the objective value of the given problem, i.e., the same as the equation (1), $C_{penalty}$ is a predefined coefficient for the penalty term, and e_j indicates the quantity of constraint violation in each constraint inequality in equation (2).

In this paper, we use the problems in OR-Library [9] and the problems generated randomly as the problem instances of the multidimensional knapsack problems. Randomly Generated Multidimensional Knapsack problems (RGMK) are generated as follows: RGMK is generated as the same way carried out by Sakawa *el al.* [10]. In the equation (1) and (2), the coefficients c_j, a_{ij} are randomly defined in the interval $[0, 999]$ in accordance with uniform distribution. Moreover, the coefficients b_i are defined by using random variable $\gamma \in [10, 20]$ and following equation:

$$b_i = \gamma \sum_{j=1}^{n} a_{ij}, \quad (i = 1, \ldots, m).$$

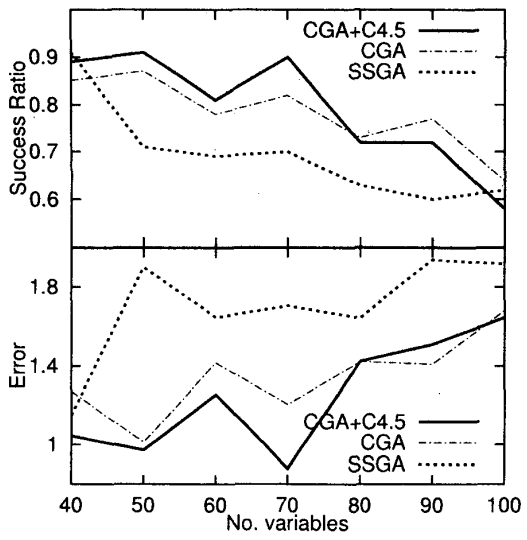

Figure 5: Experimental results on the Randomly Generated Multidimensional Knapsack Problems; (UPPER) success ratio and (LOWER) error against optimal solution with logarithm scale

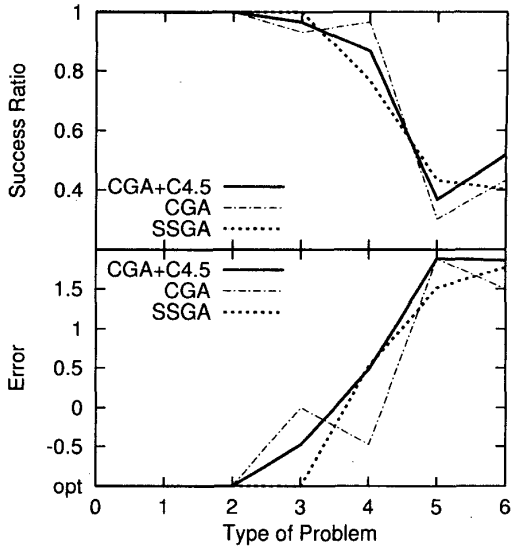

Figure 6: Experimental results on the Multidimensional Knapsack Problems in the OR-Library; (UPPER) success ratio and (LOWER) error against optimal solution with logarithm scale

In all the experimentations whose results are shown in the next subsection, we compare three kinds of evolutionary computations: Steady-State GA (SSGA) [11], conventional Coevolutionary GA (CGA), and the proposed method, i.e., Coevolutionary GA with schema extraction by C4.5 (CGA+C4.5). GA parameters of them are set to be as follows: the population size, the mutation probability, and the crossover probability of SSGA are set to be 100, 0.05, and 1.0, respectively. CGA and CGA+C4.5 have the same configuration except for using the schema extraction mechanism by C4.5 in CGA+C4.5. The H-GA in CGA and CGA+C4.5 is the same as the SSGA. The population size, the mutation probability, and the crossover probability of the P-GA are set to be 10, 0.1, and 1.0, respectively. The roulette wheel selection mechanism is adopted as the selection manner of the P-GA. Moreover, as the parameter with respect to an additional mechanism proposed in this paper, the proportions of positive instances p_p and negative instances p_n are set to be 10 and 30, respectively. The total number of classes n_c and the interval of calling C4.5 are set to be 3 and 1500.

4.2 Results

First of all, we compare the proposed method with conventional CGA and SSGA on the RGMK problems and multidimensional Knapsack problems in the OR-Library as depicted in Figure 5 and Figure 6, respectively. In Figure 5, x axis indicates the number of variables, on the other hand, x axis in Figure 6 denotes the type of problem since mknap1.txt in the OR-Library has seven problems. In these figures, y axes of upper and lower graphs denote the success ratio that means how many optimal solutions GAs can be found, and the error of objective value against the optimal solutions with logarithm scale. In these experiments, the optimal solutions of RGMK are searched by lp_solve [12] in advance, and the optimal values of the problems in mknap1.txt in the OR-Library are already known. These experiments are carried out until the number of fitness evaluations is equal to ten million (RGMK) or three million (mknap1.txt). In RGMK, 10 problem instances are generated for each variable size, 10 experiments are carried out for each problem instance, that is, the average values over 100 experiments are plotted in Figure 5, on the other hand, 30 experiments for each problem in mknap1.txt are carried out. The average values over 30 experiments are plotted in Figure 6. As

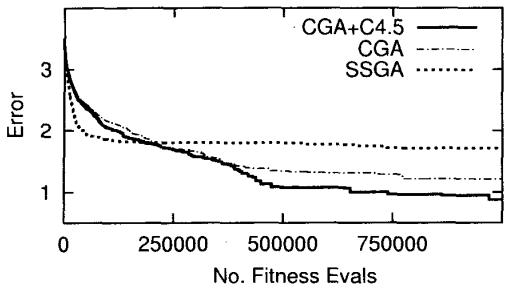

Figure 7: The temporal aspects of the error against optimal solution with logarithm scale

depicted these figures, the proposed method outperforms conventional CGA and SSGA in the case of RGMK but significant differences cannot be found in mknap1.txt.

In Figure 7, the temporal aspects of search process of the proposed method, CGA, and SSGA in the case of RGMK with 70 variables and 10 constraint inequalities are delineated. In the graph in the figure, x and y axes indicate the number of fitness evaluations and error against optimal value as well as the previous graphs'. It seems that SSGA falls into premature convergence in the figure.

Next, in order to investigate the sensitivity of parameters introduced for the proposed method, we examine various values of the parameters, namely, interval for using C4.5 I, the total number of classes n_c, and the proportion of positive and negative class p_p, p_n as delineated in Table 1, 2, and 3. In all the tables, RGMK with 10 constraint inequalities and 70 variables are examined, and the data written in bold font indicate that such parameter outperform conventional CGA against the same problem. Moreover, the underlined data denote the best performance against all parameters in the table. When the interval parameter I is more than 1000, the proposed method exhibit good performance stably. As shown in Table 3, best performance is observed, provided $p_n = 30$. The proportion parameter of positive class p_p is not affected so much.

4.3 Discussions

In the previous subsection, we examined two kinds of multidimensional knapsack problems: Randomly Generated Multidimensional Knapsack problems (RGMK) and mknap1.txt from the OR-Library. In the RGMK, the proposed method worked quite well, but significant differences couldn't be observed in the case of the mknap1.txt. As delineated in Figure 7, the proposed method allows the CGA to avoid the premature convergence. The proposed method vigorously tries to discover useful partial solutions, maybe *building blocks*, by using C4.5. The discovered schema information is incorporated into the P-GA, and is recombined by the genetic operation of P-GA. Again, effective schema information searched by the P-GA is propagated onto the H-GA population. Such cyclic processes may prevent the *hitch-hiking effects* because partial information is effectively evaluated.

I	Ratio	Fin. Sol.	I	Ratio	Fin. Sol.
10	0.82	5822.33	800	0.80	5826.57
50	**0.87**	**5832.07**	1000	**0.87**	**5830.12**
100	**0.91**	**5834.32**	1500	**0.90**	**5837.02**
300	**0.84**	5826.20	2000	**0.89**	**5832.40**
500	**0.84**	**5830.97**			

Table 1: Various results for changing parameter "interval of calling C4.5" I; some items written in bold font indicate that such parameters outperform the conventional CGA. Underlined items denote the best performance in each column

No. class	Ratio	Fin. Sol.
2	**0.85**	**5830.39**
3	**0.90**	**5837.02**
4	**0.87**	**5835.33**

Table 2: The changes of success ratio (Ratio) and final solutions (Fin. Sol.) for a variety of values of the number of classes n_c

Besides, various parameters introduced in the proposed method are examined. Experimental results tell us that the proposed method performs well, provided parameter set (I, n_c, p_p, p_n) = (1500, 3, 10, 30) which are mainly used in this paper. If the interval parameter i is set to be less value, sufficient instances of positive class and negative classes cannot be collected. However, if I is set to be larger value, enormous training set is generated, that is, it is difficult to finish C4.5 phase for reasonable computational time, because the translation process from decision tree into a set of rules may be a bottleneck in such cases.

5 Conclusion

In the framework of Coevolutionary Genetic Algorithms, we indicated that Machine Learning techniques can find effective schemata from the H-GA population by utiliz-

p_p	p_n	Ratio	Fin. Sol.	p_p	p_n	Ratio	Fin. Sol.
5	5	**0.85**	**5831.94**	10	30	**0.90**	**5837.02**
5	10	**0.85**	**5833.62**	30	5	**0.86**	**5834.14**
5	30	**0.88**	**5837.39**	30	10	**0.85**	**5833.85**
10	5	**0.86**	**5834.14**	30	30	**0.90**	**5837.02**
10	10	**0.85**	**5833.85**				

Table 3: The table for a variety of couples of p_p (proportion of positive instances) and p_n (proportion of negative instances)

ing training data which consists of a large amount of genetic information searched in H-GA and their fitness value. The extracted schema information is incorporated in natural manner into the P-GA population as schemata in the H-GA. In order to confirm the effectiveness of the proposed method, we applied the proposed method to multidimensional knapsack problems, one of combinatorial optimization problems. We could only show the preliminary results in this paper, but those results imply that Machine Learning techniques can improve the performance of Coevolutionary Genetic Algorithms.

Although we examined only for multidimensional knapsack problems in this paper, the primal idea behind the proposed method, i.e., extraction and utilization of schema information in the H-GA, can be widely adopted to various kinds of problems such as function optimization problems, combinatorial problems and so on.

This work is the first attempt for making use of Machine Learning or Data Mining techniques into our Coevolutionary Genetic Algorithm. The future works are following: (1) using other Machine Learning techniques such as CN2 [5], (2) introducing Data Mining algorithm such as Apriori [13] into CGA owing to be able to regard discovered associate rules as schemata, (3) applying the proposed method to other kinds of combinatorial problems such as Constraint Satisfaction Problems (CSPs), (4) developing a new evolutionary computation by incorporating Machine Learning techniques or Data Mining techniques into standard GAs instead of CGA.

Bibliography

[1] H. Handa, N. Baba, O. Katai, *et. al.*: Genetic Algorithm involving Coevolution Mechanism to Search for Effective Genetic Information, *Proc. of ICEC-97*, pp.709-714, 1997.

[2] H. Handa, O. Katai, N. Baba, *et. al.*: Proposal of a New Genetic Algorithm Utilizing a Mechanism of Co-Evolution (in Japanese), *Trans. of the Society of Instrument and Control Engineers*, Vol.35, No.11, pp.1438-1446, 1999.

[3] A. N. Aizawa: Evolving SSE: A Stochastic Schemata Exploiter, *Proc. of ICEC-94*, pp.525-529, 1994.

[4] R. G. Reynolds and C. Chung: Self-adaptive Approach to Representation Shifts in Cultural Algorithms, *Proc. of ICEC-96*, pp.94-99, 1996.

[5] Clark P. and Niblett T.: The CN2 Induction Algorithm, *Machine Learning*, Vol.3, No.4, pp.261–284, 1989.

[6] Cohen, W.: Fast Effective Rule Induction, *Proc. of the 12th International Conference on Machine Learning*, 1995.

[7] Quinlan J. R.: *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

[8] Goldberg, D. E. and Smith, R. E.: Nonstationary Function Optimization using Genetic Algrithms with Dominance and Diploidy, *Proc. of ICGA-87*, 1987.

[9] http://mscmga.ms.ic.ac.uk/jeb/orlib/

[10] M. Sakawa, K. Kato, T. Shibano and K. Hirose: Genetic Algorithms with Double Strings for Multidimensional Integer Knapsack Problems (in Japanese), *Journal of Japan Society for Fuzzy Theory and Systems*, Vol.12, No.4, pp.562-569, 2000.

[11] G. Syswerda: A Study of Reproduction in Generational and Steady-State Genetic Algorithms, *Foundations of Genetic Algorithms*, G. J. E. Rawlins (Eds.), Morgan Kaufmann, pp.94-101, 1990.

[12] ftp://ftp.es.ele.tue.nl/pub/lp_solve

[13] Agrwal R. and Srikant R.: Fast Algorithms for Mining Association Rules, *Proc. of the 20th VLDB Conference*, pp.487–499, 1994.