PRCAI '92

Meta-level Strategy for Genetic Algorithms based on Structured Representation

Hitoshi IBA Taisuke SATO
Machine Inference Section, Electrotechnical Laboratory
1-1-4 Umezono, Tsukuba-city, Ibaraki, 305, Japan
{iba,sato}@etl.go.jp, +81-298-58-5918

Abstract This paper presents an extension of genetic algorithms (GAs) based on structured representations. Building blocks or schemata are key notions for GA search. However classical simple GA techniques do not necessarily use these concepts for efficient search because they are not defined for the practical sake. Thus they have difficulty in effective search especially for structured spaces, such as symbolic representations. In order to avoid this difficulty, we first try to extend usual GAs to handle high-level knowledge representations, such as graphs or structured trees. The effectiveness is shown by experiments in graph acquisitions, program synthesis, and plan generations. Thereafter we introduce self-referential representations for appropriate handling of substructures in the search space. We show that GA search becomes more efficient through the use of this representation and re-usability of structured information. Finally, we attempt to establish a meta-GA control scheme based on the integration of self-referential and structured representations.

1 Introduction

The work described in this paper is one of the first results coming from a larger project concerned with knowledge acquisition and generation based on adaptive methods. As a first step, we constructed a "bug-based" adaptive learning system BUGS [Iba92a,b]. The bugs evolved by the BUGS program learn to search for bacteria in the highest density regions and are thus likely to move towards those points where the optimization function has a maximum value.

As an extension of BUGS, in this paper we attempt to realize meta-GA mechanisms. It is well known that for usual GA search, substructures such as schema or building blocks are key concepts. However, in previous approaches to GAs, these concepts are not easily incorporated for practical applications, and usually search strategy, such as crossover probability or population sizes, is decided by tried-and-error methods.

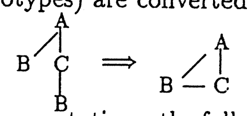
To solve this problem and realize meta-GA mechanisms, we introduce two types of representations for the extension of GAs; structured representations and self-referential representations. The structured representation (StR) is aimed at expressing high-level knowledge in the form of substructure components. The effectiveness of this representation is shown by experiments; such as graph acquisition problems and plan generations. The self-referential representation (SRR) is used for representing meta-GA control. We show the efficiency improvement through the use of SRR and discuss the reusability of self-reference. Thereafter, we integrate two representations into structured self-referential representation (StSRR) so as to develop effective GA-control mechanisms. The integrated system is intended to be applied to more practical problems, such as design problems and computer vision applications. We show how StSSR might be extended towards a meta-GA mechanism.

2 Structured representation for GAs

2.1 Operations for structured representation

Usually a string-based representation is used for GAs. However, this is not suitable for representing higher-level knowledge. [Koza90] proposed the hierarchical GA (HGA) approach, in which tree-like expressions are manipulated as gene codes for recombinations. We extend this method in order to efficiently realize graph-theoretical manipulations.

We introduce trees as basic gene codes (i.e. genotypes) for structured representations. General (simple) graphs (i.e. phenotypes) are converted from trees:



With these representations, the following graph-theoretic manipulations ($\alpha - \beta$ operations, [Lu78]) are introduced as primitive operators (Fig.1(a)).

1.	α_1	father-son splitting
2.	β_1	father-son merging
3.	α_2	brother-brother splitting
4.	β_2	brother-brother merging

We further realize the following tree-manipulations (g-operations) as recombination methods (Fig.1(b)).

5. **gmutation**	changing node label
6. **ginversion**	reordering brothers
7. **gcrossover**	replacing subtrees

These operations are regarded as natural extensions of string-based GA operators. For example, the gmutation works for avoiding the local-extrema trap just as the mutation operator in string-based GA. As can be seen in later experiments, g-operators play a role in the speed-up of convergence times. This is partly because g-operations are constructed of $\alpha - \beta$ operations as shown below.

We define the distance between trees T_1 and T_2.

$$d(T_1, T_2) := min\{\#(M) \mid M \in \{\alpha_1, \alpha_2, \beta_1, \beta_2\}^* \wedge M(T_1) = T_2\}, \quad (1)$$

where $\#(M)$ means the length of sequence of M. $M(T)$ is the tree transformed by T. Thus the distance of T_1 and T_2 is the minimum length of sequences of $\alpha - \beta$ operators, which transform T_1 into T_2. Obviously this definition satisfies the axiom of distance (i.e. $d(T,T) = 0$, $d(T_1,T_2) = d(T_2,T_1)$, $d(T_1,T_2) + d(T_2,T_3) \geq d(T_1,T_3)$). This tree distance can be derived with complexity of

$O(n^3)$ where n is the number of tree nodes [Lu89]. The effect of g-operations can be described in terms of this tree distance. For instance, in case of gcrossover, the following relations are satisfied,

$$d(T_a, T_a') = d(T_b, T_b') = d(T_1, T_1') \tag{2}$$
$$d(T_a, T_b') = d(T_b, T_a') = d(T_2, T_2'), \tag{3}$$

where T_a and T_b are the parents for the gcrossover, and T_a' and T_b' are their children. Fig.1(b) shows the subtrees T_1,T_1',T_2 and T_2' as operands of the gcrossover operator. Thus, g-operations are expected to work as useful subroutines composed of $\alpha - \beta$ operations.

Since the genetic algorithms for StR are straightforward, we omit the details. The following parameters are required for specifying the algorithms as usual GAs.

N	:	Population size
P_{GCROSS}	:	Probability of GCROSSOVER
P_{GMUT}	:	Probability of GMUTATION

2.2 Experimental results

This section gives experiments so as to show the effectiveness of graphical search using StR. In this section, we used the following parameters for all experiments.

N	:	20
P_{GCROSS}	:	0.6
$P_{GMUTATION}$	:	0.0333

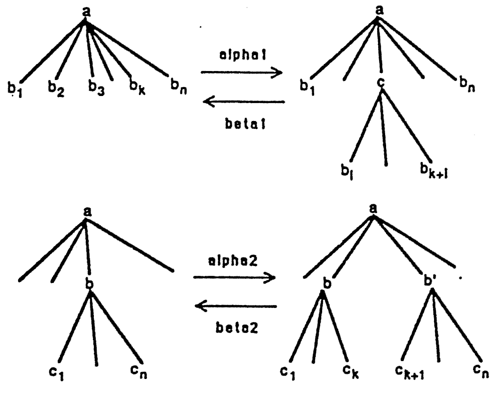


Fig.1(a) α-β operators

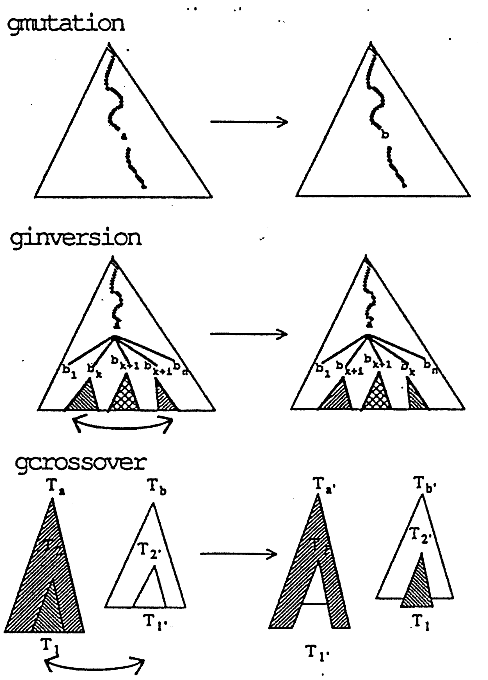


Fig.1(b) GA operators on trees

Experiment 1 Acquisitions of trees.
First, we experiment in acquisitions of trees, in which randomly generated trees evolve to a target tree (T^*). In this case, we choose the fitness of tree T as follows,

$$fitness(T) = \frac{1}{1 + d(T, T^*)}. \tag{4}$$

Fig.2 illustrates the experimental results. The first column is the target model-lists (T^*). The second column shows the fitness values (i.e. max, min, avg) at initial generations. The third shows the averaged fitness (avg) and numbers of mutations and crossovers applied so far at terminated generations (times) when the targets are successfully acquired. Fig.3 shows the process of acquisition with generation for the last target of Fig.2. As can be seen from the figure, the required generations (times) are increased according with the complexity of trees. In general, highly structured trees in graph-theoretical terms of linearity or recursion are relatively easily acquired. For the sake of comparison, the rightmost column in Fig.2 shows the result by using $\alpha - \beta$ operators only. In this case, all trees are acquired later than those results by graph operators (i.e. gcrossovers and gmutations). Especially, the difference is more remarkable for complex trees . This reflects that using only $\alpha - \beta$ operators is not effective for avoiding local-extreme trap.

Experiment 2 Collorary of Turan Claim
For an arbitrary simple graph G of n nodes, both G and its complement $\overline{G}$ have more than $\frac{n(n-1)(n-5)}{24}$ $(= \Delta(n))$ triangles in total [Lovasz79,vol.3,p.99,ex.3.32(b)]. This claim is related to the *Turan theorem.* We attempt to adaptively construct the graph of the lower bound number $\Delta(n)$ triangles. For this purpose, we use the following fitness functions:

$$fitness(G) = \frac{1}{1 + \Delta(G) - \Delta(n)}, \tag{5}$$

where $\Delta(G)$ is the number of triangles in both G and $\overline{G}$. A value of 1.0 designates the desired graph. Fig.4(a) shows the result for $n = 9$ (i.e. $\Delta(9) = 12$). In this case, the desired graph (Fig.4(b)) is adaptively acquired after 520 generations. Informally, the search space for this problem is 2^{36} (i.e. ${}_9C_2 = 36$).

Using the StR we conducted several other experiments in solving problems in graph theory. There are many invariants known for graphical features, such as Ramsey numbers, independent numbers and connectedness (See [Lovasz79] for more details). We attempted to construct a desired graph given a set of specified invariants. Some of theses problems are known to be quite difficult or unsolved. Moreover, we further experimented in higher-level knowledge acquisitions, such as functional synthesis or rule-based plan generations. The results of these experiments are encouraging and we feel the effectiveness of structured representations has been confirmed.

3 Self-referential representation

3.1 Operations for self-referential representation

This section describes the self-referential representation (SRR) introduced for the effective control of schema recombinations. In this section, we handle binary string-based genes, which will be extended to structured representations later.

The primitive operator "1-point crossover (1X)" recombines genes as follows.

$$a_1 a_2 \cdots a_j a_{j+1} \cdots a_n \implies a_1 a_2 \cdots a_j b_{j+1} \cdots b_n \tag{6}$$
$$b_1 b_2 \cdots b_j b_{j+1} \cdots b_n \implies b_1 b_2 \cdots b_j a_{j+1} \cdots a_n, \tag{7}$$

where n is the length of the chromosomes, and a_i and b_k are allele values. In this case, the crossover occurs between j-th and (j+1)-th allele. We now introduce self-reference for this basic genotype as follows.

$$a_1 a_2 \cdots a_j a_{j+1} \cdots a_n \ : \ \alpha_1 \cdots \alpha_k, \tag{8}$$

where $\alpha_1 \cdots \alpha_k$ is binary expression, whose value lies between 1 and n . We call $a_1 \cdots a_n$ the *data-part*, and $\alpha_1 \cdots \alpha_k$ the *address-part* or *reference*. Our crossover operators are applied to the point addressed by the address-parts. For instance, the crossover of

$$a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 \ : \ 0\,1\,1 \tag{9}$$

$$b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 \ : \ 1\,0\,1 \tag{10}$$

results in

$$a_1 a_2 a_3 | b_4 b_5 | a_6 a_7 a_8 \ : \ 0\,1\,1 \tag{11}$$

$$b_1 b_2 b_3 | a_4 a_5 | b_6 b_7 b_8 \ : \ 1\,0\,1 \tag{12}$$

because $011_2 = 3$ and $101_2 = 5$ in binary. The designated address may well point to self-reference itself. For instance, the crossover of

$$a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 \ : \ 1\,0\,0\,1 \tag{13}$$

$$b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 \ : \ 1\,0\,1\,1 \tag{14}$$

results in

$$a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 \ : \ 1/0\ 1/1 \tag{15}$$

$$b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 \ : \ 1/0\ 0/1 \tag{16}$$

These two types are called as non-reflective and full-reflective respectively. SRR is summarized as follows.

	Non-reflective	Full-reflective
Pointer part	Not addressed	addressed
n,m constraint	$2^{m-1} < n \le 2^m$	$2^{m-1} < n + m \le 2^m$

This SRR is intended to improve search efficiency based on schema combinations. This is realized by maintaining reference descriptions of crossovers within genes. Moreover, applying GAs to the address enables control facilities and strategic learning in the form of the meta-GA mentioned later.

Our extended GAs based on SRR is described below with the following parameters.

N	:	Population size
P_{cross}	:	Probability of crossover on data-part
P_{mut}	:	Probability of mutation on data-part
$P_{ppcross}$	:	Probability of crossover on address-part
$PPGA$	:	Period of GA on address-part

Step1 The initial population of SRRs is generated with random values and the fitness is calculated. The generation time is initialized to $t := 1$. Set up the selection counter as $n := 0$.

Step2 Select two parent P_1 and P_2 using a probability distribution over the fitness such that individuals with higher fitness are selected more frequently.

Step3 With probability P_{cross}, apply the crossover operator to the data-part of copies of P_1 and P_2, forming two offsprings C_1 and C_2. Go to Step5.

Step4 If Step3 is skipped (i.e. crossover did not occur in Step3), form two offspring C_1 and C_2 by making copies of P_1 and P_2.

Step5 If t is a multiple of PPGA, with probability P_{pcross}, apply the crossover operator to the address-parts of C_1 and C_2.

Step6 Apply the mutation operator to the two offsprings C_1 and C_2 with probability P_{mut}.

Step7 Let $n := n + 2$.

Step8 If $n < N$ then go to Step2.

Step9 Calculate fitness for each new individual. The generation counter is incremented as $t := t + 1$. The selection counter is reset as $n := 0$. Go to Step2.

In step3, the crossover point is determined by the address-parts of P_1 and P_2. As a result, the recombination is either a 1-point or a 2-point crossover. The recombination of the address-parts in Step5 is the same as that for the simple GAs so that the crossover point is decided randomly. PPGA is the learning period for address-parts, which plays an important role for non-reflective SRR. Setting this value too small might lead to falling prey to the local-extreme. Experiments show that the appropriate value for this parameter is about 20.

We now give some experimental results to show the effectiveness of our SRR approach. First, we take the exemplar fitness as,

$$fitness(x_1, \cdots, x_n) = g(x_1)^{20} g(x_2)^{20} \cdots g(x_n)^{20} \tag{17}$$

$$g(x) = 24x^2 - 22x^4 + 8x^6 - x^8. \tag{18}$$

Fig.5 illustrates the above function with n=2. This function has checkerboard-like structure, whose fields are occupied by hills and which is similar to DeJong's F5 [Booker87]. This function is sensitive to combinations by crossover. In the following experiment, we set $n = 7$ and let each x_i be in the range of -2 and 2. We decode the sequence of $x_1, x_2, \cdots, x_7$ as chromosome in binary forms and use the following parameters.

N	:	30
Chrom. length	:	50
P_{cross}	:	0.6
P_{mut}	:	0.0333
$P_{ppcross}$	:	0.4
$PPGA$	:	10

Fig.6(a) compares average and variance of the on-line performances of 24 runs with different methods at two generations (200 and 500). The performance at generations T is defined as follows [Booker87].

$$On-line(T) = \frac{1}{T}\sum_{i=1}^{T} f_E(i) \tag{19}$$

$$Off-line(T) = \frac{1}{T}\sum_{i=1}^{T} f_E^*(i), \tag{20}$$

where $f_E(i)$ is the best fitness at generations i and

$$f_E^*(i) = max\,\{f_E(1), f_E(2), \cdots, f_E(i)\} \tag{21}$$

In the method column, SGA means simple GAs. "Punctuation" is mentioned later. We use the statistical test of significance of average difference at each generation (200 and 500). Since the F-test in Fig.6(a) shows that variances are not significantly different, we can use T-test for testing average difference. In Fig.6(b) the t-test values and the test result are displayed (0.05 stands for superiority significant by 5%, n.s. for not significant, and black star for not superiority). For instance, at generations 200, non-reflective SRR is significantly superior to SGA by 5% ($3.193 > 1.960 = t_{0.05,24+24-2}$), whereas full-reflective SRR cannot be shown to be significantly superior ($0.815 < 1.960 = t_{0.05,24+24-2}$). In order to confirm that the superiority of SRR is not due to 2-point crossover (2X), we further experiment in comparing the results obtained by SRR with those obtained by random 2-point crossover. Fig.7 shows the result of on-line performances. As can be seen, the two performances are significantly different (5% significant). Thus it is suggested that by SRR approach some important information is adaptively acquired within the references. To summarize, the experimental results indicate that

1. SRR is significantly superior to SGA.
2. No difference exists between full-reflective SRR and Non-reflective SRR.

We further experimented with other functions such as DeJong's benchmark functions [Booker87] and graph-coloring problems [von Laszewski91]. For instance, Fig.8 shows the performance data for DeJong F5. Through these experiments, the advantage of SRR is confirmed, in spite of the different convergence for different problems. However, it should be noted that the superiority of SRR

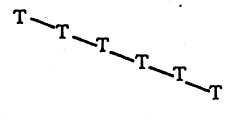
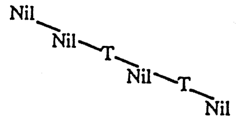
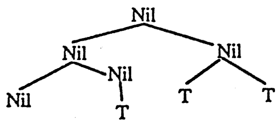
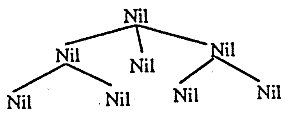
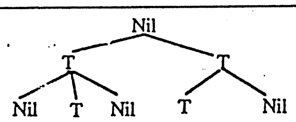
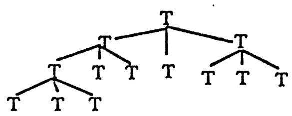
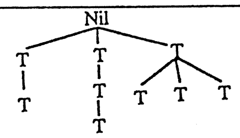
model-list random-tree with node=(T, NIL)	initial		success times			$\alpha-\beta$ only at	times
	*max*	0.2000	6	*avg*	0.3342	*max*	0.2500
						$n\alpha_1$	40
	min	0.0200		*nmu*	20	$n\alpha_2$	41
						$n\beta_1$	32
	avg	0.1433		*ncr*	32	$n\beta_2$	9
	*max*	0.2500	15	*avg*	0.3933	*max*	0.5000
						$n\alpha_1$	85
	min	0.1250		*nmu*	42	$n\alpha_2$	76
						$n\beta_1$	98
	avg	0.1790		*ncr*	85	$n\beta_2$	21
	*max*	0.2000	20	*avg*	0.3188	*max*	0.2000
						$n\alpha_1$	115
	min	0.0120		*nmu*	93	$n\alpha_2$	109
						$n\beta_1$	122
	avg	0.1433		*ncr*	120	$n\beta_2$	27
	*max*	0.1429	38	*avg*	0.3420	*max*	0.1667
						$n\alpha_1$	229
	min	0.0015		*nmu*	167	$n\alpha_2$	243
						$n\beta_1$	201
	avg	0.0350		*ncr*	166	$n\beta_2$	103
	*max*	0.2000	50	*avg*	0.3197	*max*	0.1667
						$n\alpha_1$	344
	min	0.0047		*nmu*	337	$n\alpha_2$	351
						$n\beta_1$	361
	avg	0.0572		*ncr*	365	$n\beta_2$	153
	*max*	0.1667	82	*avg*	0.3406	*max*	0.1000
						$n\alpha_1$	453
	min	0.0213		*nmu*	736	$n\alpha_2$	522
						$n\beta_1$	550
	avg	0.0979		*ncr*	488	$n\beta_2$	225
	*max*	0.1667	98	*avg*	0.3401	*max*	0.1667
						$n\alpha_1$	608
	min	0.0185		*nmu*	599	$n\alpha_2$	584
						$n\beta_1$	579
	avg	0.1146		*ncr*	592	$n\beta_2$	278

Fig.2 Experimental results for tree acquisitions

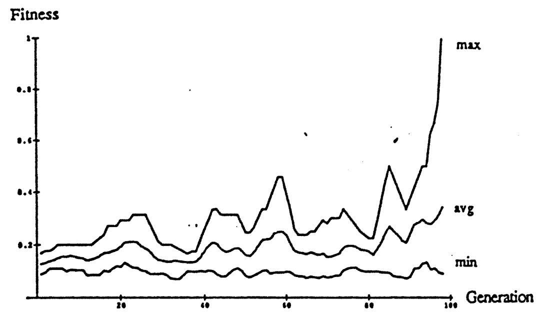


Fig.3 Fitness changes with generations

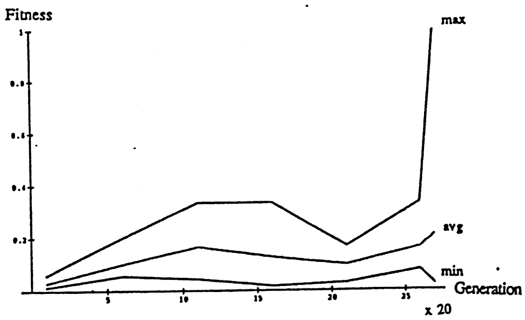


Fig.4(a) Fitness changes with generations

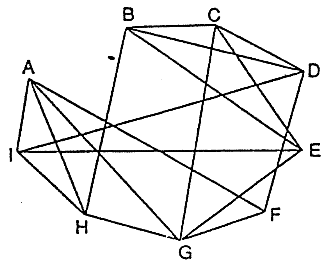


Fig.4(b) Acquired graph

appears especially in highly structured space such as the above fitness (Fig.5). On the contrary, significant advantage is not observed for non-structure functions, such as DeJong F4, in which random Gaussian noise is added at each point. For efficient search by GAs, an appropriate handling of schemata is required. We think that the SRR approach gives a method for strategic control of schemata within address-parts. In order to ascertain this effect, we further experiment in inserting the acquired address parts into another randomly initialized population. Fig.9 shows the result for the problem of Fig.5, in which three types of address parts are given to the initial population; random address parts, acquired address parts at generations 500 and 1000. As was expected, initial performance is significantly different. This suggests that essential information is stored within address parts for an appropriate handling of schemata.

4 Extensions and discussions

Although initial results of experiments with our StR and SRR approaches are encouraging, several points need to be addressed. This section discusses the integration of the two approaches so as to develop the StSRR (Structured Self-Referential Representation) for extended GAs.

4.1 Related work

Improvement of efficiency for GAs is studied by several researchers ([Louis91] [Shaefer87]). For instance, "punctuation" is introduced as a tag in [Schaffer87]. This representation realizes the adaptive uniform crossover. For example, the crossover of

$$a_1a_2a_3a_4a_5a_6a_7a_8 \;:\; 1\,0\,0\,1\,0\,0\,1\,0 \quad (22)$$
$$b_1b_2b_3b_4b_5b_6b_7b_8 \;:\; 1\,0\,0\,1\,0\,0\,0\,1 \quad (23)$$

results in

$$b_1b_2b_3a_4a_5a_6b_7a_8 \;:\; 1\,0\,0\,1\,0\,0\,1\,0 \quad (24)$$
$$a_1a_2a_3b_4b_5b_6a_7b_8 \;:\; 1\,0\,0\,1\,0\,0\,0\,1, \quad (25)$$

where the crossover points are represented in the tag. Crossover points are derived from those positions whose values are changed form 0 to 1 or from 1 to 0 when reading the tag from left to right. The "punct" row in Fig.6 shows the results obtained by this method. As can be seen, the punctuation is not significantly superior to any other method (5% significant). This is due to the fact that the punctuation method puts too much emphasis on short-length schemata.

4.2 Representational gap for GAs

It should be noted that remarkable effects are quite dependent on features of the search space. As mentioned earlier, a highly structured space is more suitable for our SRR approach. Moreover, the effectiveness of SRR relies on gene coding (how to transform genotypes into phenotypes). In general, the gray coding is more suitable than the binary coding for usual optimizations [Caruana88]. This is because the gray coding benefits from the crossover of substructures more than the binary coding; overall fitness of schema is maintained and gradually increased through search in the gray coding, whereas

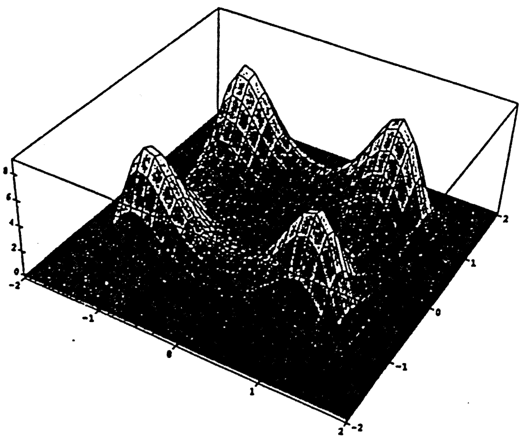


Fig.5 Fitness function

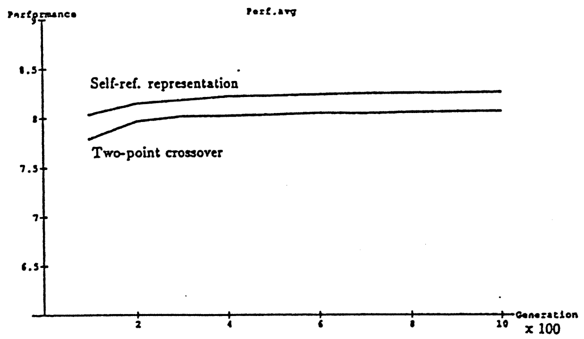


Fig.7 Comparison with two-point crossover operator

Method	Generation:200		Generation:500	
	mean	variance	mean	variance
SGA	8.011	0.026	8.059	0.006
Punctuation	7.944	0.022	7.997	0.007
Non-ref. address	8.148	0.017	8.226	0.007
Ref. address	8.118	0.014	8.213	0.006

(a)

Method	count	*SGA*		*Punct.*		*Non. adr.*		*Ref. adr.*	
SGA	24	——		1.487	(ns.)	-3.193	(∗∗∗)	-2.586	(∗∗∗)
Punct.	24	-1.487	(∗∗∗)	——		-4.973	(∗∗∗)	-4.412	(∗∗∗)
Non. adr.	24	3.193	(0.05)	4.973	(0.05)	——		0.815	(ns.)
Ref. adr.	24	2.586	(0.05)	4.412	(0.05)	-0.815	(∗∗∗)	——	

(b)

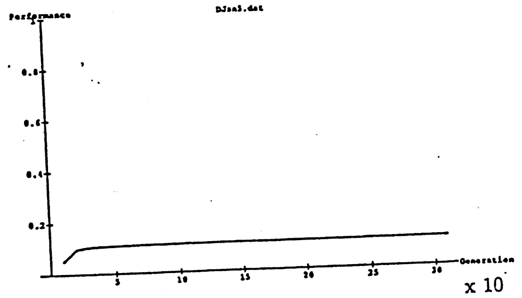
Method	count	*SGA*		*Punct.*		*Non. adr.*		*Ref. adr.*	
SGA	24	——		2.580	(0.05)	-7.038	(∗∗∗)	-6.657	(∗∗∗)
Punct.	24	-2.580	(∗∗∗)	——		-9.168	(∗∗∗)	-8.850	(∗∗∗)
Non. adr.	24	7.038	(0.05)	9.168	(0.05)	——		0.551	(ns.)
Ref. adr.	24	6.657	(0.05)	8.850	(0.05)	-0.551	(∗∗∗)	——	

(c)

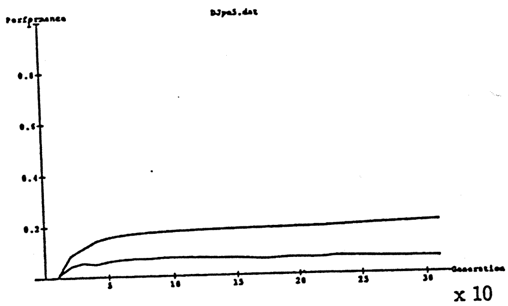
Fig.6 Experimental results and T-test

```
Population size (popsize)          = 60
Chromosome length (lchrom)         = 34
Maximum # of generation (maxgen)   = 300
Crossover probability (pcross)     = 0.6000
Mutation probability (pmutation)   = 0.0333
Reflexive GA prob. (ref-prob)      = 1.0000
Punctation GA peirod (pga-period)=   20
Initial Punct. prob. (initpunctp)= 0.5000
Default Pcrossover prob.(ppcross)= 0.4000
Default Pmut prob. (pmutation)     = 0.0333
```

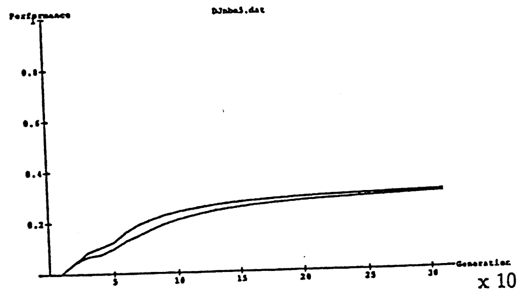
(a) GA parameters



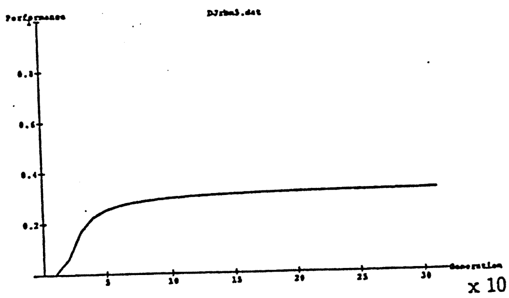
(b) SGA [on-line and off-line performance (coinciding)]



(c) Punct. [on-line (bottom), off-line (top)]



(d) Non. Adr. [on-line (bottom), off-line (top)]



(e) Ref. Adr. [on-line and off-line performance (coinciding)]

Fig.8 Experimental results of DeJong's function No.5

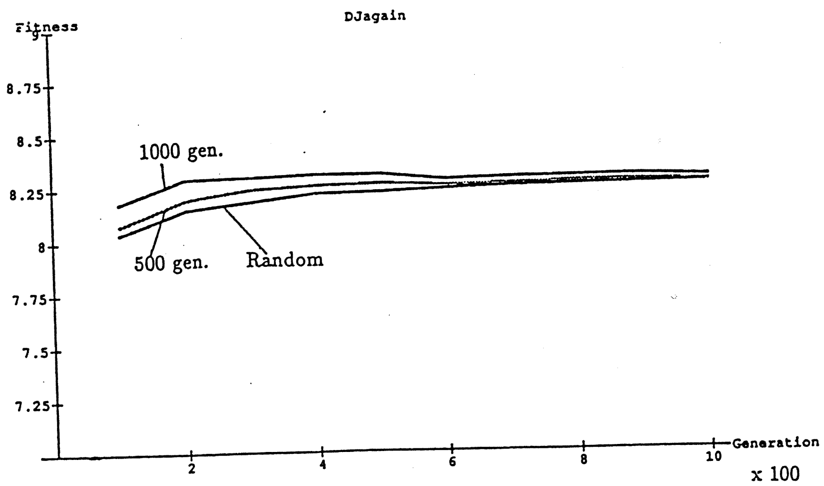


Fig.9 Experimental results of re-use of references

too drastic changes occur by the crossover from the binary coding (i.e. Hamming Cliff). In the experiments described in Section 3 we observed that the SSR's effectiveness is less remarkable for the gray coding than the binary coding. Thus, SSR approach makes compensation for the representational gaps through the use of the appropriate scheme control by address-parts. Our current research inevitably leads to important future concerns, such as representational changes and adaptive representations for GAs, some of which are described in the next section.

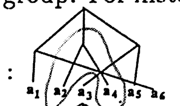
4.3 Meta-level strategy learning for GAs

In order to realize effective GA control mechanisms, we integrate our two methods; StR and SRR. The basic idea is to maintain the scheme structures of search spaces in reference expressions, which enables effective meta-level control. For this purpose, we extend self-referential expressions into structured forms as follows.

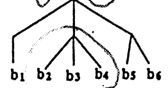
$$a_1a_2\cdots a_ja_{j+1}\cdots a_n \; : \; Tree(a_1a_2\cdots a_ja_{j+1}\cdots a_n), \tag{26}$$

where $a_1a_2\cdots a_ja_{j+1}\cdots a_n$ means a usual gene. In address part, $Tree(a_1a_2\cdots a_ja_{j+1}\cdots a_n)$ is a tree whose terminal nodes are labeled as $a_1a_2\cdots a_ja_{j+1}\cdots a_n$. We call this integrated representation as StSRR (Structured Self-Referential Representation). When applying crossover, the branch of the tree is handles as a group. For instance, the crossover of

$$\mathbf{P_1} \quad a_1a_2a_3a_4a_5a_6 \; : \tag{27}$$

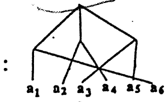


$$\mathbf{P_2} \quad b_1b_2b_3b_4b_5b_6 \; : \tag{28}$$

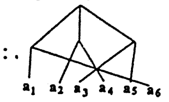


results in

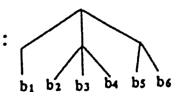
$$\mathbf{C_{1a}} \quad a_1b_2a_3b_4a_5a_6 \; : \tag{29}$$



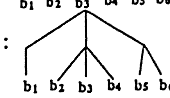
$$\mathbf{C_{1b}} \quad b_1a_2b_3a_4b_5b_6 \; : \tag{30}$$



$$\mathbf{C_{2a}} \quad b_1a_2a_3a_4b_5b_6 \; : \tag{31}$$



$$\mathbf{C_{2b}} \quad a_1b_2b_3b_4a_5a_6 \; : \tag{32}$$



In this case, the address part of P_1 is applied to produce C_{1a} and C_{1b} because a_1,a_6,a_2,a_4 and a_3,a_5 are treated as subgroups for the uniform crossover. As a result, four children are reproduced, two of which are selected in the later selection phase.

The benefit of this StSRR can be explained as follows. It is claimed that in usual GAs long-length schemata (for instance, $a_1 * * * * a_6$ in P_1) are frequently destroyed by combinations [Louis91]. This is because the crossover point will more frequently lie between a_1 and a_6. On the contrary, with P_1 in StSRR, a_1 and a_6 are always treated as a group and this schema is never destroyed by crossover. Moreover, the learnability in structured reference part is guaranteed by the effectiveness of StR as described in Section 2.

StSRR's version of GAs is given by modifying STEP4 and STEP6 in SSR's GAs as follows.

STEP4: Apply Graph GA in section 2 onto address part.
STEP6: Select N individuals out of 2N children according to their fitness distribution.

To show the effectiveness of this StSRR approach, we conducted several experiments, such as order-3 deceptive problems [Goldberg90] and computer vision applications [Iba92a,b]. These results and further improvements will be addressed in subsequent works.

5 Conclusion

This paper described two representational approaches for expressing extended GAs; structured representation for higher-level knowledge and self-referential representation for appropriate handling of schemata by search. The effectiveness of our approaches is shown by experiments, such as solving graph-theoretic problems and optimizing benchmark functions. Thereafter we integrated these two approaches to realize a strategic learning for GA search based on structured self-referential representation.

Our important next step is to establish meta-GA control mechanisms. We currently investigating this topic based on adaptive representations and representational changes.

Acknowledgment.
We have profitted from many suggestions and discussions with members of our section and laboratory. Comments on a draft of this paper by Thomas Weigert are gratefully acknowledged. We also thank him for helping us to a improve the English of this paper.

References

[Booker87] Booker, L. Improving search in genetic algorithms, In *Genetic algorithms and simulated anneling*, (ed. Davis, L.), pp.61–73, Morgan Kaufmann, 1987

[Caruana88] Caruana, R. A. and Schaffer, J. D. Representation and hidden bias: gray vs. binary coding for genetic algorithms, In *Proc. of 5th Machine Learning*, 1987

[Goldberg89] Goldberg, D. E. Genetic algorithms in search, optimization and machine learning, Addison Wesley, 1989

[Goldberg *et al.*90] Goldberg, D. E. Deb, K. and Korb, B. Messy genetic algorithms revisited: studies in mixed size and scale, In *Complex systems*, vol.4, 1990

[Iba *et al.*91] Iba, H. and Inoue, H. Reasoning of geometric concepts based on algebraic constraint-directed method In *Proc. of 12th IJCAI*, pp.143–149, 1991

[Iba *et al.*92a] Iba, H. Akiba, S. Higuchi, T. and Sato, T. BUGS : A bug-based search strategy using genetic algorithms, ETL-TR92-8, to appear in *Proc. of 2nd PPSN*, 1992

[Iba *et al.*92b] Iba, H. deGaris, H. and Sato, T. Bug-based search strategy and problem solving, ETL-TR92-24, submitted to *SAB92*, 1992

[Koza90] Koza, J. Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems, Report No. STAN-CS-90-1314, Dept. of Computer Science, Stanford Univ., 1990

[Louis91] Louis, S. J. and Gregory J. E. Rawlins Designer genetic algorithms: genetic algorithms in structure design, In *Proc. of International Conference on Genetic Algorithms*, 1991

[Lovasz79] Lovasz, L. Combinational problem and exercises, Akademiai Kiado, Budapest, 1979

[Lu78] Lu, S. and Fu, K. Error-correcting tree automata for syntactic recognition, In *IEEE Tr. on computers*, vol.c-27, no.11, 1978

[Lu84] Lu, S. A tree-matching algorithm based on node splitting and merging, In *IEEE Tr. on pattern analysis and machine intelligence*, vol.PAMI-6, no.2, 1984

[Schaffer87] Schaffer, J. D. An adaptive crossover distribution mechanism for genetic algorithms, In *Proc. of International Conference on Genetic Algorithms*, 1987

[Shaefer87] Shaefer, G. G. The ARGOT strategy: adaptive representation genetic optimizer technique, In *Proc. of International Conference on Genetic Algorithms*, 1987

[von Laszewski91] von Laszewski, G. Intelligent structural operators for the k-way graph partitioning problem, In *Proc. of International Conference on Genetic Algorithms*, 1991