Solving Constraint Satisfaction Problems by a Genetic Algorithm Adopting Viral Infection

Hitoshi Kanoh,
Kazuyo Hasegawa, Miyuki Matsumoto,
Nobuko Kato and Seiichi Nishihara

August 30, 1996

ISE-TR-96-137

Keyword: *Genetic algorithm, Constraint satisfaction problem, Evolution, Virus, Infection, Search, Selection, Population.*

Institute of Information Sciences and Electronics
University of Tsukuba
Tsukuba, Ibaraki 305, Japan
Phone: 0298-53-5344, E-mail: kanoh@is.tsukuba.ac.jp

Solving Constraint Satisfaction Problems by a Genetic Algorithm Adopting Viral Infection

Hitoshi Kanoh, Kazuyo Hasegawa,
Miyuki Matsumoto and Seiichi Nishihara
Institute of Information Sciences and Electronics
University of Tsukuba

Nobuko Kato
Tsukuba College of Technology

Abstract

Several approximate algorithms have been reported to solve large constraint satisfaction problems (CSPs) in a practical time. While these papers discuss techniques to escape from local optima, this paper describes a method that actively performs global search. The present method is to improve the rate of search of genetic algorithms using viral infection instead of mutation. Partial solutions of a CSP are considered to be viruses, and a population of viruses is created as well as a population of candidate solutions. Search for a solution is conducted by crossover. Infection substitutes the gene of a virus for the locus decided by the virus. Experimental results using randomly generated CSPs prove that the proposed method is faster than a usual genetic algorithm(GA) to find a solution when the constraint density of a CSP is low.

1. Introduction

Constraint satisfaction problems (CSPs) are search problems to assign discrete values to variables so as to satisfy all given constraints [1]. Several approximate algorithms based on min-conflicts heuristic [1] have recently been proposed to solve large CSPs in a practical time (MCHC [2], GSAT [3], the breakout algorithm [4], GENET [5], EFLOP [6], and so on). Techniques to escape from local optima have

been discussed in these papers, but there are very few methods that actively perform global searches [7][8].

The authors have proposed a hybrid search method that combines a genetic algorithm (GA) with min-conflicts hill-climbing (MCHC [2]) taking into account the global search ability of GA [8]. Detailed experiments have also been performed to prove that the proposed method gives higher probability of finding a solution than randomly restarting MCHC, the when constraint density of a CSP is so low as to become stuck at locally optimal points. The rate of search of the method, however, did not improve for such CSPs. In this paper, we propose a method to improve the rate of search of the GA which is a part of the hybrid GA described in reference [8].

Search in usual GAs, based on neo-Darwinian evolutionary theory, is conducted by crossover and mutation [9]. We plan to improve the rate of search by giving direction to evolution, that is search, using crossover and viral infection. Anderson pointed out the evolutionary significance of viral infection in the journal Nature [10]. Nakahara et al. reported the mechanism of organic evolution by viral infection and its superiority to mutation [11]. So the subjects of this paper are to propose an effective definition of a virus and an algorithm of infection for CSPs.

In this paper we address CSPs whose constraints are expressed by sets of partial solutions (defined by Haralick [1]). Partial solutions of the CSP are considered to be viruses, and a population of viruses is created as well as a population of individuals as candidate solutions. Infection substitutes the genes of a virus for individual's loci decided by the virus. In the following sections, we first define the problem to be addressed and explain our strategies. Second we define the virus for CSPs and describe a role of viral infection as a search method as well as the algorithm of the proposed method. Finally, we show experimental results to prove that the proposed method is faster than a usual GA to find a solution when the constraint density of the CSP is low.

2. Strategies

2.1. Objective problem

CSP is a general term of a problem whose whole character can be defined by local constraints among the components of the problem. Layout problems, timetabling problems, Boolean satisfiability problems and graph coloring problems are typical

examples of CSPs. In this paper we address the CSP of which constraints are expressed by sets of partial solutions defined by Haralick [6]. Problems to reconstruct 3-D models of polyhedrons from three orthographic views of mechanical components are examples of this type of CSPs.

A CSP is defined by a quadruple (U,L,T,R). Figure 1 shows an example of a CSP. Here, $U = \{X_1,\cdots,X_n\}$ is a set of variables in a given problem, where n is the number of variables. The term $L = \{a,b,c,\cdots\}$ is a finite set of values, and each member corresponds to a candidate of a value that should be assigned to each variable. The term $T = \{T_1,\cdots,T_m\}$ is a set of constraint relations of variables where m is the number of constraints, and each $T_j = (p,q)$ means that there is a constraint between the p-th variable and the q-th variable. And $R = \{R_1,\cdots,R_m\}$ is a set of R_j, and each member R_j is a set of partial solutions that can be assigned to variables (X_p,X_q) expressed by T_j. For example, $R_j = \{(r_1,r_2),(r_3,r_4),\cdots\}$ means $(X_p,X_q) = (r_1,r_2)$ or $(r_3,r_4)\cdots$. In this paper we address only binary constraints.

No general effective algorithms for solving CSPs exist, as it is NP-complete to decide whether a CSP has a solution or not. In order to consider effective algorithms, CSPs are often classified into several types according to the relationship between m and n [2][4][8]. In this paper we address the CSPs so that the size of the search space is large and constraint density is low; in concrete terms, the former is 10 to the 30-th power and the latter is less than or equal to 5. This type of CSP is considered to be hard to solve using the representative exact algorithm (DP procedure [13]) or approximate algorithm (MCHC).

$U = \{X_1, X_2, X_3, X_4\}$,
$L = \{a, b, c, d, e\}$,
$T = \{T_1, T_2, T_3, T_4\}$,
$T_1 = (1,2)$, $T_2 = (1,3)$, $T_3 = (3,4)$, $T_4 = (1,4)$,
$R = \{R_1, R_2, R_3, R_4\}$,
$R_1 = \{(a,b),(e,c)\}$, $R_2 = \{(a,c),(a,e)\}$, $R_3 = \{(c,e),(a,c)\}$,
$R_4 = \{(a,d),(a,e),(b,c)\}$.

Solution : $(X_1,X_2,X_3,X_4) = (a,b,c,e)$.

Figure 1. An example of a CSP

2.2. Solving CSPs using GAs and problems

GAs are a general search method [9]. They use analogs of genetic operators on a population of states in a search space to find those states that have high fitness values. In optimization problems, genetic operators and coding methods are designed in advance so that the individuals may satisfy the constraints. In contrast, the objective of CSPs is to find an individual that satisfies the constraints by regarding the number of satisfied constraints as the fitness value. Figure 2 shows, as an example, the solution of CSP in Fig. 1 which was encoded on the GA. We will henceforth define the directional search, to select a variable and assign a value to the variable according to some plan, and discuss the efficiency of search in GAs using it.

GAs are based on the neo-Darwinian evolutionary theory; search is conducted by selecting individuals which have fewer constraint violations, that is to say conflicts, from those individuals generated by crossover and mutation. Crossover is generally considered a robust search means. Offspring may inherit partial solutions without conflict from their parents, but no information to decide which genes are partial solutions is available. From a search-strategic point of view, this means that variables are randomly selected and then values which will be assigned to them are also randomly selected from genes contained within a population. Therefore search by crossover in GAs can not be considered efficient. Furthermore, mutation is a random search method itself. When we think about efficiency of global search that is the most important characteristic of GAs; we must admit that the rate of search is low, because GAs stress random search rather than directional search.

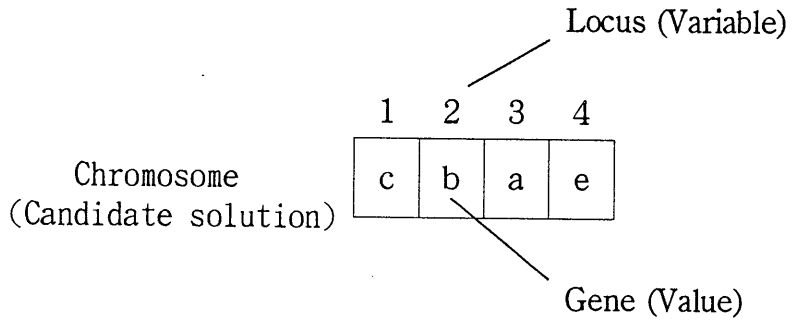


Figure 2. Coding of a CSP (UL code)

It can be considered that the above problem is directly inherited from problems of the neo-Darwinian evolutionary theory. In our study, we planed to improve the rate of

search by basing the virus theory of evolution [11] summarized in the next section.

2.3. The virus theory of evolution

Nakahara and Sagawa have proposed the evolutionary assumption that the cause of evolution is viral infection [14]. Viruses can transfer genes from one organism to target ones, and may change chromosomes of them. In the neo-Darwinian evolutionary theory, it has been considered that the organisms evolve as follows; the fittest characters for survival are produced by mutation, and these characters are inherited to offspring. However, the probability that mutated dominant individuals appear and increase in nature is very low. Nakahara explains this problem in Ref. [11].

> We believe that vertical gene transfer through viral infection to a reproductive cell, as well as horizontal gene transfer through viral infection among individuals, occurs in this mechanism. We do not claim that this theory can clarify all issues pertaining to evolution. Nevertheless, our new viral theory of evolution does resolve four points not fully explained by Darwinism: 1) the rapid evolution of species, 2) the rapid extinction of species, 3) evolution progressing in a specific direction and 4) parallel segregation taking place.

2.4. Strategies

Figure 3 shows the main procedure of the proposed search method based on the virus theory of evolution. The procedure is the same as usual GA's except for using viral infection instead of mutation. We will henceforth call this method GAVI (genetic algorithm with viral infection). Our strategies are as follows.

(1) A population of viruses is created in addition to a population of individuals.
(2) Partial solutions of CSPs are considered to be viruses.
(3) Infection substitutes the genes of a virus for individuals' loci decided by the virus.

Figure 4 shows an example of infection for the candidate solution in Fig. 2. The first element (a,b) of R_1 in Fig. 1 is regarded as a virus, and loci 1 and 2 of the individual are substituted for a and b, respectively, according to $T_1=(1,2)$.

```
procedure main(){
   initialize a population of Np individuals;
   initialize a population of viruses;
   set infectivity for the viruses;
   for generaton = 1 to Ng{
      calculate the fitness values of the individuals;
      selection;
      crossover;
      infection();
   }
}

procedure infection(){
   select Np' individual at random;
   for k = 1 to Np' {
      select the virus which has a high infection rate;
      substitute the gene of the virus for loci of the k-th individual;
      renew the fitness value of the individual;
      renew the infectivity of the virus;
   }
}
```

Figure 3. The algorithm of the proposed method

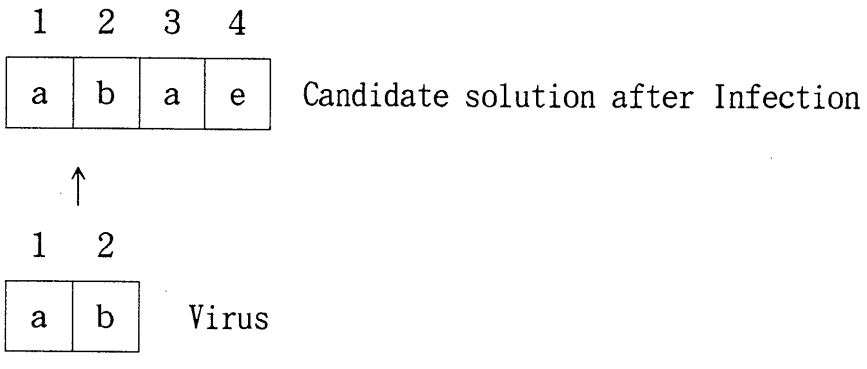


Figure 4. An example of infection

3. Proposed method

3.1. Coding of a CSP

Let the gene at the i-th locus in the chromosome of the k-th individual in the population be denoted by $G_k(i)$. The variable and value of CSP are usually encoded on the GA as a locus and a gene respectively (see Fig. 2), that is,

$$G_k(i) \in L \quad (i=1,\cdots,n) \tag{1}$$

where i corresponds to a variable number. We will henceforth call this encoding UL code, and define UL-search to search for values which should be assigned to variables using UL code.

Another encoding method is as follows: constraint relation T_j and set of partial solutions R_j are encoded as locus and allele respectively, that is,

$$G_k(j) \in R_j \quad (j=1,\cdots,m) \tag{2}$$

where j corresponds to a constraint number. Figure 5 shows an example of the encoding method for the candidate solution in Fig. 1. We will henceforth call this encoding TR code, and define TR-search to search for the elements of R_j which should be assigned to T_j using TR code.

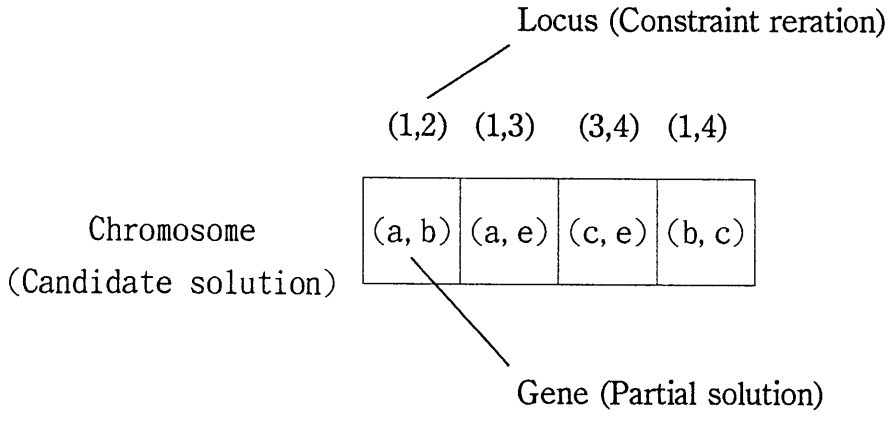


Figure 5. Coding of a CSP (TR code)

3.2. Definition of a Virus

We define the population of viruses by the following expression (3),

$$V = \bigcup_{j=1}^{m} W_j = W_1 \cup \cdots \cup W_m \qquad (3)$$

where $W_j = \{(T_j, y) \mid y \in R_j\}$.

When all the elements of V are numbered at random and written as $V_1, \cdots, V_N \in V$; V_r (r=1,···,N) is defined as a virus. As an example, V for the CSP in Fig. 1 as follows.

$$V = \{ ((1,2),(a,b)), ((1,2),(e,c)), ((1,3),(a,c)), ((1,3),(a,e)), ((3,4),(c,e)), ((3,4),(a,c)), ((1,4),(a,d)), ((1,4),(a,e)), ((1,4),(b,c)) \} \qquad (4)$$

This means that genes in TR code are regarded as viruses, so viral infection is merely TR search. The present method (GAVI) expresses a candidate solution by UL code and searches for a solution using viral infection instead of mutation in a usual GA. Consequently, GAVI can be a hybrid method which combines UL-search, that is crossover, with TR-search that is infection.

3.3. Parameter

In this section we define infection rate and infectivity as parameters of GAVI. First, infection rate is a parameter which determines the ratio of the number of crossovers to the number of infections, and defined as follows. Infection rate: the number of infected individuals with viruses divided by the number of created individuals by crossover. Also, infection rate means the ratio of the number of trials for UL-search and TR-search.

Table 1. The effect of infection rate and infectivity on search strategy

	UL-search	TR-search
Means	Selection	Infection
Direction	Fitness of individual	Infectivity of virus
Number of trials	Np × Ng	Np × Ng × Infection rate

Np:Number of individuals, Ng:Number of generations

We next set infectivity for all viruses. Let viruses be selected to infect individuals according to infectivity. Let the infectivity increase (decrease) as the result of rewriting a chromosome of an individual with a virus when increasing (decreasing) the fitness value of the individual. Infectivity is a parameter which determines the direction of TR-search, and corresponds to fitness of an individual in UL-search. Table 1 shows the effect of infection rate and infectivity on search strategy.

3.4. Algorithm

Figure 3 shows the algorithm of the proposed method, where the population size, the number of viruses and the upper bound of generations are denoted by Np, Np'and Ng, respectively. In Fig. 3, first, Np' viruses are randomly generated as well as Np individuals, then the following operations are repeated until a solution is found or Ng times.

(1) Calculation of fitness

Fitness values for all individuals are calculated using the following expression [8].

$$F_k = 1 - \sum_{j=1}^{m} CONF_k(j) / m$$

where $T_j = (p,q)$, and m is the number of constraints,

$$CONF_k(j) = \begin{cases} 0 & (G_k(p),G_k(q)) \in R_j \\ 1 & \text{otherwise} \end{cases}$$

(2) Selection and crossover

These operations are the same as usual GAs.

(3) Infection

First, Np' individuals are randomly selected from a population of Np individuals, where the infection rate equals Np'/Np. Next, the procedure after the "for" statement in the procedure infection in Figure 3 is executed for all selected individuals.

4. Experiments

4.1. Experimental method

To evaluate the performance of the proposed method, we randomly generated 1500 CSPs and tried to solve them under the following conditions,

$|U| = n = 50$

$|L| = 4$

$|T| = m = 49, 100, 150, 200, 250$

(namely $d = 0.98, 2, 3, 4, 5$)

$|R_{mean}| = 2, 4, 6$

where $|R_{mean}| = \sum_{j=1}^{m} |R_j|/m.$

The following two values were examined on a HP 712/60 workstation, and all algorithms were implemented in C language.

(a) Percentage of success (%): The number of solved CSPs divided by the number of searched CSPs $\times$ 100.

(b) Mean solution time (sec): The mean time required to find a solution with respect to successful cases.

Roulette selection, where the selection probability was fitness squared, with conserving the elite was used in the experiments. And uniform crossover was also used. In addition, the mutation rate per locus of the GA was fixed at 2% as a result of preparatory experiments.

4.2. Experimental results

(1) Infection rate

Table 2 shows experimental results for GAVI on those CSPs which have the lowest constraint density (d=0.98) in the range of the infection rate = 0 to 60%. Table 2 leads to the following: the percentage of success drops when infection is not conducted, but is about the same when the infection rate is greater than 5%, then the

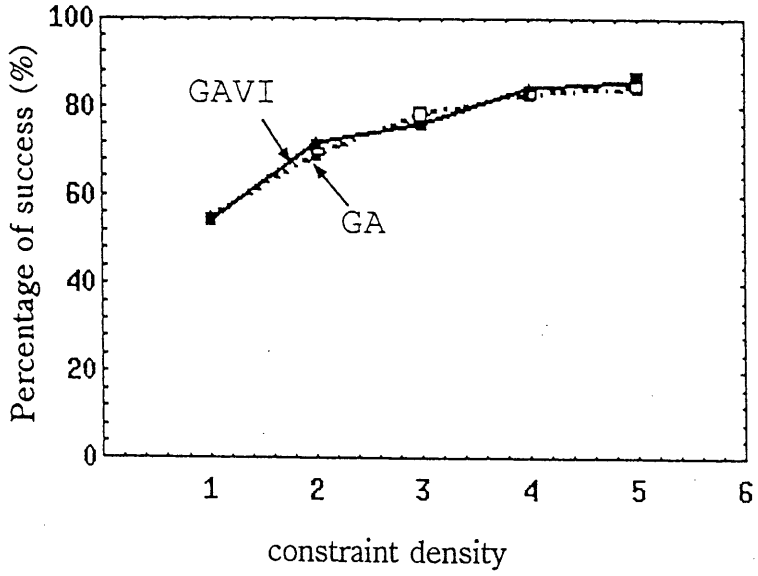
mean solution time is also about the same. Infection rate is a peculiar parameter of GAVI. It can be considered that exact optimization for the parameter is not necessary while adjustment of parameters is generally difficult in approximate algorithms.

Table 2. Experimental results for infection rate

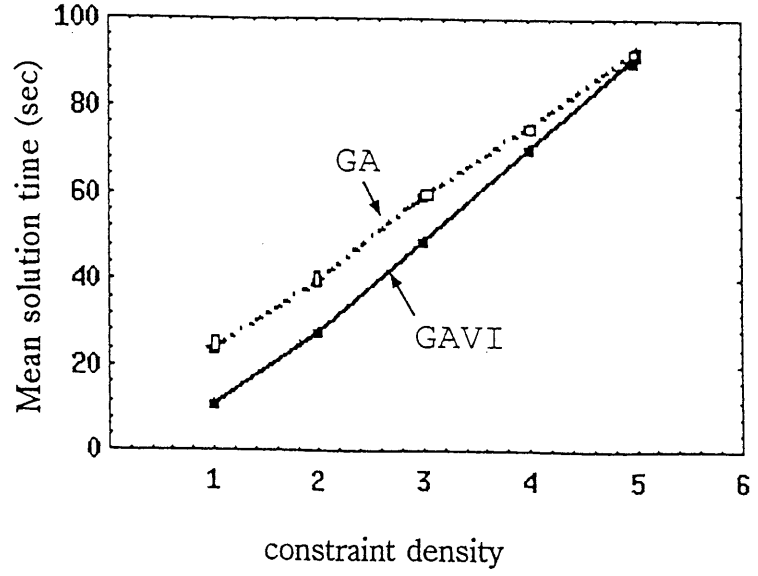
Infection rate (%)	0	1	3	5	10	20	60
Percentge of success (%)	43	49	50	56	56	56	57
Mean solution time (sec)	9	12	12	13	12	11	12

(2) Comparison of GA and GAVI

Figure 6 shows the results, where the infection rate is equal to 40%. . The transverse axis is the constraint density defined by d = m/n; it is known that CSPs of d<5 are hard to solve. Each point is the average of 300 CSPs. The following are seen in Fig. 6: comparing GA with GAVI, the percentage of success is about the same over the entire region; and the mean solution time is also about the same when d = 4 to 5; in contrast, the proposed method is 1.5 to 2 times faster than GA in the range of d = 0.98 to 2.



(a) The number of solved CSPs divided by the number of searched CSPs × 100.



(b) The mean time required to find a solution.

Figure 6. Experimental results for GA and GAVI

4.3. Discussion

In this section we will discuss why the present method (GAVI) has a greater rate of search than GA. We first consider size of search space as a index showing the degree of difficulty to find a solution. Let S_{UL} and S_{TR} be the sizes of search spaces for UL-search and TR-search, respectively. They can be expressed as follows,

$$S_{UL} = |L|^n$$

$$S_{TR} = \prod_{j=1}^{m} |R_j| \sim |R_{mean}|^m = (|R_{mean}|^d)^n.$$

Here, if we define α as follows, we can be obtained that UL-search is more efficient than TR-search when $\alpha \ll 1$ for given CSPs, and when $\alpha \gg 1$ the contrary is true.

$$\alpha = (S_{UL}/S_{TR})^{1/n} \sim |L|/|R_{mean}|^d \qquad (4)$$

Table 3 shows the average value of α at $|R_{mean}| = 2$, 4 and 6 for each constraint density in Fig. 6.

Table 3. Averaged α and constraint density d

d	0.98	2	3	4	5
$\overline{\alpha}$	1.2	0.45	0.19	0.09	0.043

As GAVI is a hybrid search method that combines UL-search with TR-search, it may be natural that GAVI has a greater rate of search than GA which conducts only UL-search when $\alpha > 1$ (i.e. d=0.98 in Fig. 6(b)). In spite of $\alpha < 1$, however, GAVI is faster than GA when $d \geq 2$. This reason is to introduce infectivity of a virus showing the direction which should be searched. While GA decides the direction by only fitness of an individual, GAVI uses both fitness and infectivity. As the result of this, the domain which should be searched may be limited, and the time required to find a solution may become small.

It has been generally considered that it is difficult to find a complete solution which satisfies all constraints using GA, because their ability of local search is not so high. GAVI has the same problem as this, so the percentage of success shown in Fig. 6(a) is not so high. In addition, as we mentioned in the first chapter, we proposed GAVI on the assumption that GAVI was combined with MCHC which is a representative local search method.

5. Conclusions

In this paper we proposed a solution to constraint satisfaction problems using viral infection instead of random mutation in genetic algorithms. We performed experiments using randomly generated CSPs and showed that the mean time required to find a solution is better than usual GAs. In this paper we addressed the CSP of which constraints are expressed by sets of partial solutions defined by Haralick. Problems to reconstruct 3-D models of polyhedrons from three orthographic views of mechanical components are examples of this type of CSPs. We consider that the proposed method is also effective for the following CSPs: problems whose parts of constraints are expressed by the set of partial solutions; and problems whose partial solutions are easy to find, though a complete solution is hard to obtain. The advantage of the present method is that the rate of search can be improved as the result of combining two different coding methods. This result, it seems, may be applied to many problems. In our forthcoming study, we will combine the present method with a local search method and perform detailed comparison with other methods. We are also considering to applying it to practical problems such as 3-D reconstructing and real-time path planning.

Acknowledgments

The authors wish to thank Dr. Hideomi Nakahara of Yamanashi Medical College and Mr. Takashi Sagawa for their considerable assistance with the virus theory of evolution, and we also wish to thank Prof. Tutomu Hoshino of the University of Tsukuba for his helpful advice in carrying out this research.

This research was supported in part by Grant-in-Aid for Scientific Research (C) of the Ministry of Education, Science, Sports and Culture, Grant No. 08680384.

References

[1] R. M. Haralick, L. G. Shapiro. The Consistent Labeling Ploblem, Part I. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. PAMI-1, No. 2, pp.173-184, 1979.

[2] S. Minton, et al.. Minimizing conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems. *Artificial Intelligence*, 58, pp.161-205, 1992.

[3] B. Selman, H. Kautz. Domain-Independent Extension to GSAT : Solving Large Structured Satisfiability Proglems. *AAAI'93*, pp.290-295 (1993).

[4] P. Morris. The Breakout Method For Escaping From Local Minima. *AAAI'93*, pp.40-45, 1993.

[5] A. Davenport, E. Tsang, C.J. Wang, K. Zhu. GENET: A Connectionist Architecture for Solving Constraint Satisfaction Problems by Iterative Improvement. *AAAI'94*, pp.325-330, 1994.

[6] N. Yugami, Y. Ohta, H. Hara. Improving Repair-based Constraint Satisfaction Methods. *AAAI'94*, pp.344-349, 1994.

[7] J. Bowen, G. Dozier. Solving Constraint Stisfaction Problems Using A Genetic/Systematic Search Hybrid That Realizes When to Quit. *ICGA–6*, pp.122-129, 1995.

[8] H. Kanoh, M. Matsumoto, S. Nishihara. Genetic Algorithms for Constraint Saticfaction Problems. *IEEE Proc. of SMC'95*, pp.626-631, 1995.

[9] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison Wesley, 1989.

[10] N.G. Anderson. Evolutionary significance of virus infection. *Nature*, 227, pp.1346-1347, 1970.

[11] H. Nakahara, T. Sagawa, T. Fuke. Virus theory of evolution. *Bulletin of Yamanashi Medical College*, Vol.3, pp.14-18, 1986.

[12] K. Uchino, H. Kanoh, S. Nishihara. Understanding three orthographic views by using constraint knowledge base. *Journal of Japanese society for artificial intelligence*, Vol.11, No.3, pp.157-165, 1996.

[13] D. Mitchell, B. Selman, H. Levesque. Hard and Easy Distributions of SAT Problems. *AAAI'92*, pp.459-465, 1992.

[14] Kinji Imanishi. Private letter. 1971.