Genetic Algorithms for Constraint Satisfaction Problems

Hitoshi Kanoh,
Miyuki Matsumoto, Seiichi Nishihara

November 30, 1995

ISE-TR-95-126

This report is a revision of the following paper:
IEEE International Conference on Systems, Man, and Cybernetics
Volume 1 of 5, pp.626-631, October 1995, Vancouver.

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

Genetic Algorithms for Constraint Satisfaction Problems

Hitoshi Kanoh, Miyuki Matsumoto and Seiichi Nishihara
University of Tsukuba
Tsukuba, Ibaraki 305, Japan

ABSTRACT

Several approximate algorithms using hill-climbing techniques and neural networks have recently been proposed to solve large constraint satisfaction problems (CSPs) in a practical time. In these proposals, many methods of escaping from local optima are discussed, however, there are very few methods that actively perform global search. In this paper we propose a hybrid search method that combines the genetic algorithm with the min-conflicts hill-climbing (MCHC). In our method, the individual that has the fewest conflicts in the population is used as the initial value of MCHC to search locally. A detailed experimental simulation is also performed to prove that the proposed method generally gives better efficiency than randomly restarting MCHC when CSPs are sparsely-connected.

Key words: Genetic algorithm, hill climbing, global search, local minima, Hamming distance, constraint satisfaction, conflict.

1. INTRODUCTION

A constraint satisfaction problem (CSP) is a combinatorial search problem to find solutions that satisfy given constraints [1]. A decision problem, whether a CSP has solutions or not, is NP-complete, and no general effective algorithm to solve it exists. Several approximate algorithms that use iterative improvement have been studied to solve large scale CSPs with a high probability of finding a solution as much as possible in a practical time. While exact algorithms using backtrack techniques satisfy completeness of the algorithm, approximate algorithms do not satisfy it, but may perform high-speed and highly parallel processing.

Good results have particularly been obtained by hill-climbing in the direction of minimizing the number of conflicts (MCHC) [2] and a modified discrete Hopfield network (GDS) [3]. Because of the possibility of becoming stuck at locally optimal points, however, CSPs – where the number of constraints is closer to the number of variables – are hard to solve using these methods [4]. Techniques to escape from local optima have also been

proposed (GSAT [5] [6], the breakout method [7], GENET [8], EFLOP [9] and so on), but there are very few methods that actively perform global searches.

On the other hand, Genetic Algorithms (GAs) that have a global search strategy are a class of randomised search algorithms based on a model of organic evolution, and have been used successfully to solve optimization problems in many fields [10]. However, there are very few reports using GAs to solve CSPs. This seems to be due to a lack of local search techniques in GAs. While quasi-optimal solutions to optimization problems may be obtained, at least one complete solution that satisfies all constraints is needed in CSPs.

In this paper we propose a hybrid search method that combines GA with MCHC. Global search using GAs is performed, and the elite, or the individual with the fewest conflicts in the population, is used as the initial value of MCHC to search locally. For the purpose of giving the first priority in the GA process to maintaining diversity of the population, fitnesses of individuals are decreased when the Hamming distances between the elite and these individuals is small. The elite falling into local optima is removed from the population, and the MCHC restarts from another initial value that is created by GA.

In the following sections, we first classify CSPs into four types according to the relationship between the number of variables and constraints. Second, we describe a method to code CSPs for genetic search as well as the proposed hybrid method. Finally, we compare it with the repeated restarting MCHC from random initial values using CSPs defined by Haralick [1].

2. CLASSIFICATION OF CSPs

2.1 CSPs and Their Classification

Boolean satisfiability problems and graph coloring problems have been used as examples to study CSPs. In this paper we take Haralick's general definition [1] of CSPs to consider applicability of our method. A CSP is defined by a quadruple (U,L,T,R). Here, $U = \{1,\cdots, n\}$ is a set of variables where n is the number of variables, and each member corresponds to a component of a given problem. The term L is a finite set of values, and each member corresponds to a candidate of a value that should be assigned to each variable. $T = \{T1,\cdots, Tm\}$ is a set of constraints relations of variables where m is the number of constraints, and

the meaning of $Tj = (p,q)$ is that there is a constraint between the p–th variable and the q–th variable. $R = \{R1,\cdots,Rm\}$ is a set of Rj, and each member Rj is a set of partial solutions that can be assigned to variables in Tj. The following shows an example of a CSP.

$U = \{1, 2, 3, 4\}$

$L = \{a, b, c, d, e\}$

$T = \{T1, T2, T3, T4\}$

$T1 = (1,2)$ $T2 = (1,3)$ $T3 = (3,4)$ $T4 = (1,4)$

$R = \{R1, R2, R3, R4\}$

$R1 = \{(a,b),(e,c)\}$ $R2 = \{(a,c),(a,e)\}$ $R3 = \{(c,e),(a,c)\}$ $R4 = \{(a,d),(a,e),(b,c)\}$

Solution : (1,2,3,4) = (a,b,c,e)

In this paper only binary constraints are treated. A fast algorithm using heuristics has been proposed [1] for these CSPs described by partial solutions. In this paper, however, we will discuss them using only the number of conflicts without problem specific heuristics, as we study algorithms for general CSPs.

Because a CSP is NP–complete, no general effective algorithm for it exists. So, classifying CSPs into the following four types according to the relationship between m and n, and considering effective algorithms for each type is our standpoint.

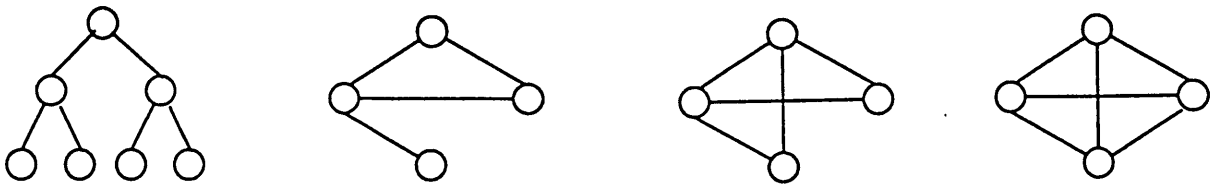
Type 1 : $m = n-1$

Type 2 : $n-1 < m << n(n-1)/2$

Type 3 : $n-1 << m < n(n-1)/2$

Type 4 : $m = n(n-1)/2$

Figure 2 shows examples of the four types of CSPs that are expressed by the constraint graphs, where the vertices and the edges correspond to the variables and the constraints, respectively. The constraint graph of type 1 is a tree structured graph and constraint density, $d = m/n$, is minimum, while that of type 4 is a complete graph having a maximum value of d.



Type 1 Type 2 Type 3 Type 4

Figure 1 Classification of CSPs

Generally, it is difficult to solve a CSP that has small d, as it has many local optima. This is the case for type 1. The n-queens problem, a standard benchmark for testing search algorithms, corresponds to type 4. Sparsely connected graphs for the graph coloring problem in reference [4], which is often referred to as a representative approximate algorithm to solve CSPs, correspond closely to type 1. The densely connected graphs correspond to type 3. It seems that the appropriate application of each algorithm is made clear by the above classification.

2.2 Approximate Algorithms to Solve CSPs

Several approximate algorithms using MCHC have recently been proposed to solve large-scale CSPs in a practical time ([2], [5], [7], [9]). MCHC uses the following heuristic as a value ordering rule [2].

Min-conflicts heuristic: Select a variable that is in conflict, and assign it a value that minimizes the number of conflicts.

Good performance can be obtained by MCHC and GDS. A drawback of these methods is a local optima problem, and a number of remedies have been proposed (GSAT, GENT and so on). In this paper we compare genetic algorithms with almost the same method as GSAT and discuss the results. That is, when the number of conflicts does not decrease even if the hill-climbing step is repeated a given number of times, it is regarded as local optima, and the procedure restarts with another randomly generated initial value. We will henceforth call this method iterated hill-climbing (IHC). IHC is based on the following hypothesis about a form of cost function on state space.

Restarting hypothesis: Solutions are far from the local optima on the CSP of which the constraint density is low.

2.3 Genetic Algorithms

GAs use analogs of genetic operators [10], that is to say selection, crossover and mutation, on a population of states in a search space to find states without conflict. GAs are based on the following hypothesis to search globally, and it is considered that different characteristics from the hill-climbing methods and neural networks can be obtained.

Building blocks hypothesis: Individuals that have few conflicts include many partial solutions. The probability that partial solutions are combined to form better individuals by genetic operations is higher than the probability that they are broken.

3. PROPOSED METHOD

3.1 Strategies

In this paper we propose a hybrid search method that combines GA with MCHC. We will henceforth call this method GAHC. Our strategies are as follows.

(1) First, a global search using the GA is performed. Then the elite, or the individual with the fewest conflicts in the population (break ties randomly), is used as the initial value of MCHC to search locally.

(2) The highest priority in the GA process is given to maintaining diversity of the population so as to enhance the ability of the global search. For that purpose, let the fitness value of the individual decreased, as the Hamming distances from the elite become small.

(3) The elite falling into local optima is removed from the population. Then the GA process is resumed, and MCHC restarts from a new elite. This procedure is based on the restarting hypothesis.

3.2 Coding and Fitness

In our model CSPs are encoded on the GA as shown in Table 1. Let the chromosome of the k–th individual in the population be denoted by Gk, and the gene at the i–th locus in the chromosome be denoted by $Gk(i)$. The third row in Table 1 shows an example of the CSP in.

Table 1 Correspondence between CSP and GA

CSP	GA	Example
Variable	Locus i	1, 2, 3, 4
Value	Gene $G_k(i)$	a, b, c
Candidate of solution	Chromosome G_k	(a, b, a, c)

Let the fitness value Fk be calculated as follows.

$$Fk = 1 - \sum_{j=1}^{m} CONFk(j) \,/\, m \qquad (1)$$

where $Tj = (p,q)$,

m is the number of constraints,

$$CONFk(j) = \begin{cases} 1 & (Gk(p), Gk(q)) \text{ not within } Rj \\ 0 & \text{otherwise} \end{cases}$$

3.3 Probability of Selection

As we mentioned above, let the probability of selection be decreased for those individuals that are close to the elite. Let Ge be the chromosome of the elite, and let the Hamming distance between Ge and Gk be defined by the following equation (2) here.

$$Hk = n - \sum_{i=1}^{n} \delta\,(Gk(i), Ge(i)) \qquad (2)$$

where n is the number of loci,

$$\delta\,(x,y) = \begin{cases} 1 & x=y \\ 0 & \text{otherwise} \end{cases}$$

Using this Hamming distance, let the probability of selection be calculated as follows.

$$PSk = Fk \times Hk \,/\, n \qquad (3)$$

3.4 Algorithm

Figure 3 shows the algorithm of the proposed GAHC, where the population size and the upper bound of generations are denoted by Np and Ng, respectively. In Figure 3, first, Np individuals are randomly generated, then the following two procedures are repeated until a solution is found or Ng times.

Local search: Fitness values for all individuals are calculated using equation (1). The individual with a maximum fitness value is the elite. If the fitness value of the elite Fe is greater than the threshold value Ft, then a local search is performed by MCHC as the elite is an initial value.

Genetic operations: First, *Nd* individuals are selected according to *PSk* in eq.(3) using a roulette selection with conserving the elite, that is including the elite to the next generation. Next, *Np* individuals are generated from them using a uniform crossover. The last operation is a mutation, that is, a random gene is assigned at a random locus in a random individual in terms of the given probability.

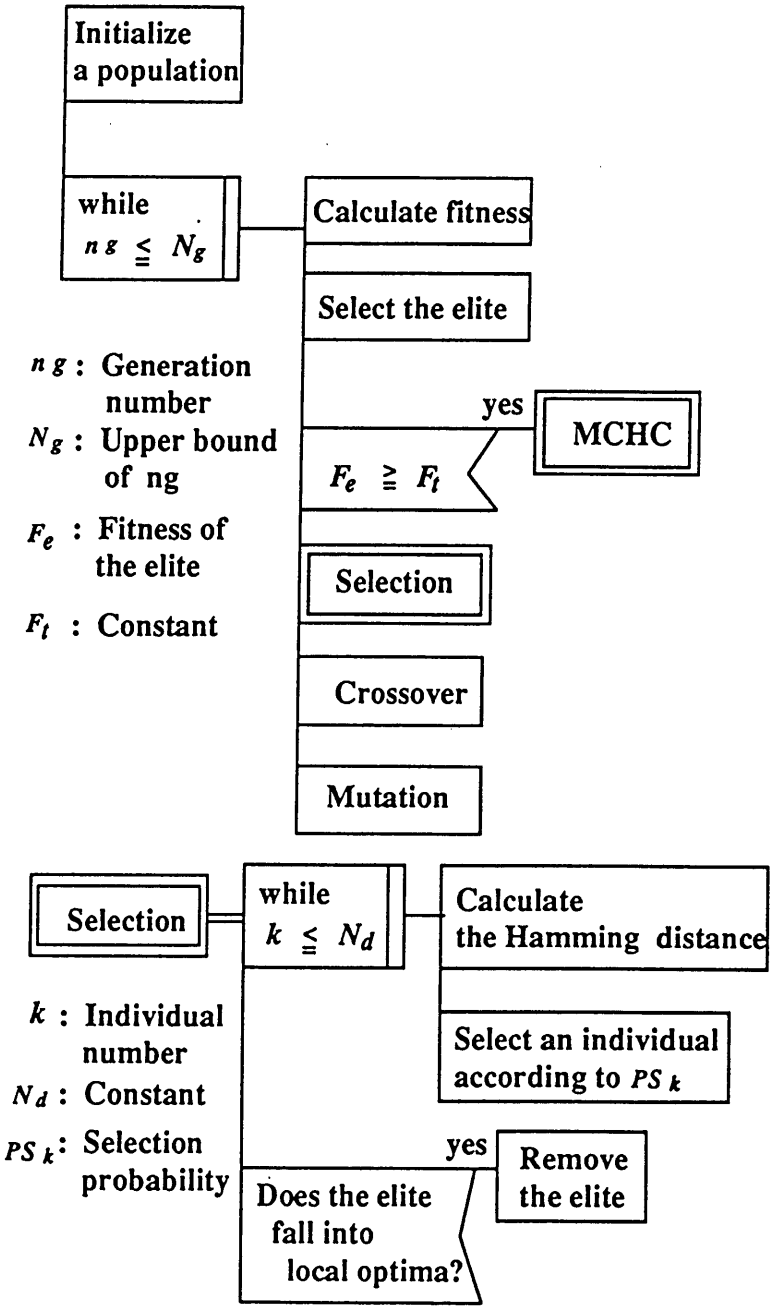


Figure 2 GAHC Algorithm

4. EXPERIMENT

4.1 Experimental Method

To evaluate the performance of GAHC, we randomly generated type 1 to 4 CSPs, and tried to solve them using IHC, GA and GAHC. This section shows the results, where the number of variables and values are fixed at 50 and four, respectively; therefore the size of the state space we treat is about 10 to the 30–th power. Table 2 shows the number of constraints and their density for each type. The following two values were examined on a HP 715/33 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 × 100.

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

Table 2 Number of constraints and their density for Type 1 to 4 CSPs

Type of CSP	1	2	3	4
Number of constraints : m	n-1	9(n-1)	17(n-1)	n(n-1)/2
Constraint density : d=m/n	0.98	8.82	16.7	24.5

n:Number of variables

4.2 Setting of Parameters

It is necessary to take notice of the comparison of GA with IHC, as they are based on quite different search strategies. The statistic values of the percentage of success and mean solution time may be changed according to *Ni* and *Ng*, where *Ni* is the upper bound of the number of iterations in IHC. In this paper we have set *Ni* and *Ng* so that the theoretical maximum values of the calculation costs can be equal between IHC and GA. These costs can be calculated as follows.

$$COST(IHC) = C_{HC} \times Nh \times Ni \quad (4)$$

$$COST(GA) = C_{GA} \times Np \times Ng \quad (5)$$

$$COST(GAHC) = C_{GA} \times Np \times Ng + C_{HC} \times Nh \times Ng \quad (6)$$

where

$C_{HC} = C1 \times m \times n$, $C_{GA} = C2 \times m + C3 \times n$

C1, C2, C3 : constants

C_{HC} is the cost to calculate one hill-climbing step, and is in proportion to the cost to calculate the number of conflicts for every variable. C_{GA} is the cost to calculate one generation per individual in the GA, and is the sum of the following two terms. The first term is the cost to calculate the number of conflicts for an individual, and is in proportion to m (cf. eq.(1)). The Second term is the sum of the calculation cost of PSk and genetic operations, and is in proportion to n (cf. eqs.(2) and (3)). Table 3 shows experimental values of C_{HC}/C_{GA}. Here, Nh is the mean number of actually doing a hill-climbing step.

Table 3 Experimental values of C_{HC}/C_{GA}

Type	1	2	3	4
C_{HC}/C_{GA}	2.98	9.37	14.5	15.8

We are very interested in comparing GAHC with GA and IHC on type 1 CSPs. In the following experiments, therefore, we estimated equations (4), (5) and (6) using the value of type 1 in Table 3, and set each parameter so that the following condition can be satisfied.

$$COST(GAHC) \leqq COST(HC),\ COST(GA) \qquad (7)$$

In addition, the survival rate Nd/Np is equal to 0.5, and the mutation rate per locus is equal to 1% throughout the experiments.

4.3 Experimental Results for GA

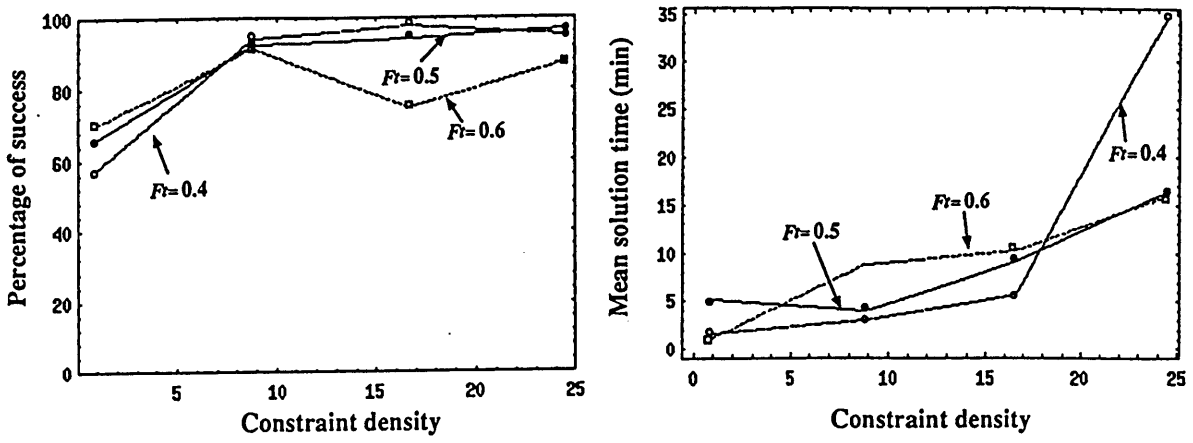
The roulette selection, where $PSk = Fk^2$, without conserving the elite and the uniform crossover are used in this experiment. Throughout the experiment, Ng is equal to 200. To compare GA on these conditions, the GA with conserving the elite has also been tested. Table 4 shows the results averaged over 1,600 CSPs in each case. Each case includes the same number of type 1 to 4 CSPs. It is seen in Table 4 that the best performance has been obtained for GA without the elite, where Np = 2000.

Table 4 Experimental results for GA

	GA without elite			GA with elite		
Population size : Np	1000	2000	3000	1000	2000	3000
Percentage of success (%)	72	77	81	74	74	73
Mean solution time (min)	9.0	8.9	12	7.1	11	20

4.4 Experimental Results for GAHC

The results are shown in Figure 4, where Np = 1000 and Ng = 100 are the set values in this experiment. The horizontal axis shows the density of constraint defined by $d = n / m$, and corresponds to different types of CSPs as shown in Table 2. Each point shown, is the average of 40 CSPs. Ft in Fig. 4 is a threshold value of the elite's fitness that implies the transfer point from the global search to the local search. This is the most important parameter in GAHC. The performance of GAHC should agree with IHC at Ft = 0 and with GA at Ft = 1. Fig. 4(a) shows that the percentage of success drops at d = 16.7, when Ft = 0.6, and Fig. 4(b) shows that the mean solution time increases rapidly at d = 24.5, when Ft = 0.4. Therefore, the best results are obtained when Ft = 0.5.



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

(b) The mean time required to find a solution

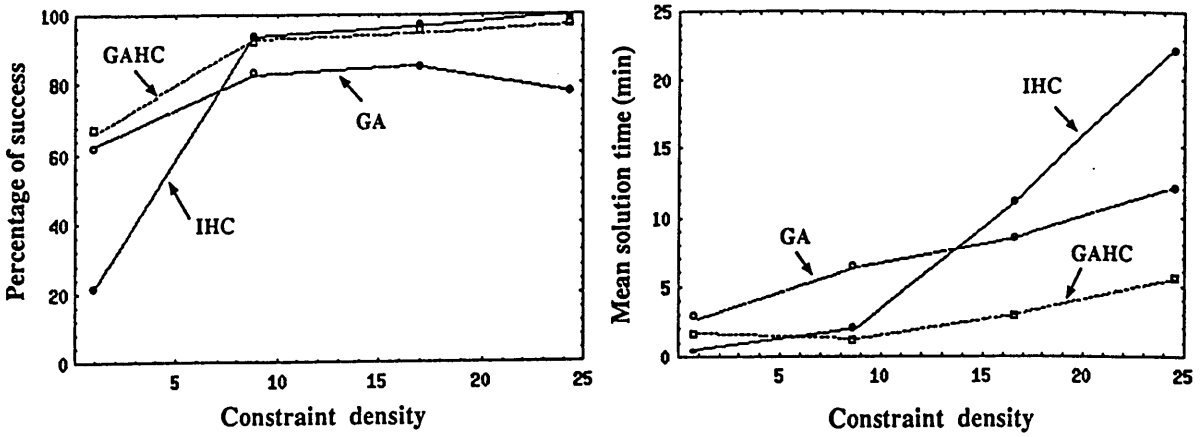
Figure 3 Experimental results for GAHC

4.5 Comparison of GAHC and other Methods

Figure 5 shows the results for GAHC, GA and IHC on type 1 to 4 CSPs. Each point shown, is the average of 400 CSPs. In this experiment, Ni = 500 is used for IHC, Nh = 100 for GAHC and Nh = 200 for IHC. Their measurement are taken from actual type 1 CSPs. Using these parameters, the expression (7) can be satisfied.

The following are seen in Fig. 5. First, comparing IHC with GAHC, the percentage of success is about the same when d = 8 to 25. In addition, GAHC is excellent when d = 0.98, where a global search strategy becomes especially important in solving CSPs. The

mean solution time is about the same when $d < 9$. In contrast, GAHC is four times faster than IHC in the range of d = 16 to 25. Moreover, while IHC increases rapidly in the range of d = 9 to 25, GAHC increases slowly. Next, comparing GA with GAHC, the latter is excellent over the entire region in Fig. 5(a) and (b).



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

(b) The mean time required to find a solution

Figure 4 Experimental results for IHC, GA and GAHC

5. CONCLUSIONS

In this paper we proposed a hybrid search method that combines the genetic algorithm with the min-conflicts hill-climbing. We performed experiments using randomly generated CSPs defined by Haralick, and showed that the probability of finding a solution and the mean solution time for the proposed method are better than iterated hill-climbing and the usual genetic algorithm.

Acknowledgment

The authors wish to thank Dr. Li Jiang Hong and Dr. Kanji Uchino of University of Tsukuba for their beneficial discussions in carrying out this research.

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, 1979, pp. 173-184.

[2] S. Minton, M.D. Johnston, A.B. Philips,P. Laird, "Solving Large-Scale Constraint Satisfaction and Sceduling Problems Using a Heuristic Repair Method", *AAAI'90*, 1990, pp. 17-24.

[3] H.M. Adorf, M.D. Johnston, "A Discrete Stochastic Neural Network Algorithm For Constraint Satisfaction Problems", *IJCNN'90*, 1990, III pp. 917-924.

[4] S. Minton, et al., "Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems", *Artificial Intelligence*, 58, 1992, pp. 161-205.

[5] B. Selman, H. Levesque, D. Mitchell, "A New Method for Solving Hard Satisfiability Ploblems", *AAAI'92*, 1992, pp. 440-446.

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

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

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

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

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