Coevolutionary searching method by double-layered population with genetic information exchange

Hisashi Handa*, Osamu Katai*, Norio Baba+ and Tetsuo Sawaragi*

*Department of Precision Engineering, Graduate School of Engineering, Kyoto University, Yoshida-Hommachi, Sakyo-ku, Kyoto 606-8501, Japan.
E-mail: {handa, katai, sawaragi}@prec.kyoto-u.ac.jp
+Department of Information Science, Osaka Educational University, Kashihara, Osaka 582-8582, Japan.
E-mail: baba@is.osaka-kyoiku.ac.jp

Abstract

Two kinds of coevolutionary genetic algorithms involving two populations with layered structures are introduced. The lower layer populations in our algorithms are used to search for optimal or satisfiable solutions of given problems like traditional GAs, and the upper layer populations are used to search for and store useful genetic information that are expected to augment and accelerate the search abilities of the lower layer populations, i.e., accelerate schema formation at the lower layers. Also, new genetic operators for exchanging genetic information between these two layers are introduced yielding effective and efficient search of our coevolutionary GAs.

In the former coevolutionary GA, the upper layer discovers and stores useful schemata which are then transmitted to the lower layer population by two genetic operators, superposition and transcription, which will be used to overwrite the individual genetic information in the lower layer population. The latter coevolutionary GA is confined to constraint-oriented problem solving. In this coevolutionary GA, the upper layer discovers and stores unsuccessful schemata by utilizing a traditional problem solving method in constraint-oriented problem solving such as solution synthesis algorithms called invasion algorithm. This negative information is then transmitted to the lower layer GA and is used to lower the fitness values of individuals that are expected to arrive at unsatisfiable solutions. In other words, the above information is used to inhibit schema formation that are expected to mislead the evolutionary processes in the lower layer, i.e., to evade form premature convergence.

Furthermore, we introduce a difficulty index of constraint satisfaction problems and that of partial solutions by referring to Shannon's information content which are then used to elucidate the progress of schema formation. Finally, these coevolutionary GAs are applied to constraint satisfaction problems such as graph coloring problems elucidating the effectiveness and the characteristics of our coevolutionary genetic algorithms.

Keywords
Genetic Algorithm, Coevolution, Constraint Satisfaction Problem, Layered Approach

1. Introduction

The notion of Constraint Satisfaction is one of the common general frameworks for problem solving in Artificial Intelligence. A constraint satisfaction problem, more precisely a consistent labeling problem consists of the elements in the problem and the constraint relationships among these elements [1], [2]. This notion of problem formulation being universal and simple, it is quite important to devise general problem solver for CSPs (Constraint Satisfaction Problems). Also, it is practically important to devise effective CSP solvers. Recently, many researchers have tried to solve CSPs by Genetic Algorithms. GAs have been widely applied to engineering problems, for instance, scheduling problems, robot control, manufacturing design problems and so on. By using GAs, it is expected to discover optimal or near optimal solutions by population-based search.

In this paper, we devise a double layered solutions space of CSP, that is, solution-level and schema-level spaces. The lower layer population explores the solution-level space, and the upper layer population explores the schema-level space.

First, an evolutionary computation with layered populations which consist of two levels, i.e., solution-level and schema-level is proposed. The solution-level population searches for good solutions in a given problem. The schema-level population searches for good schemata in the solution-level population. These populations are evolved by Genetic Algorithms [3], [4], [5], [6] and affect with each other by two genetic operators to exchange genetic information. Namely, in this case, coevolution by layered populations brings about improving search ability of GA. Further, we will introduce new evolutionary computation which consists of a Genetic Algorithm coupled with Seidel's invasion algorithm. The invasion algorithm
is one of solution synthesis algorithms commonly used to solve CSPs. This algorithm can find all satisfiable solutions in given problems, but it necessitates enormous computational load. In this paper, we use a pseudo-invasion algorithm to synthesize higher-order solutions from lower-order partial solutions, which has an ability to detect illegal high-order solutions that may cause premature convergence in GA. Furthermore, we introduce an index measuring the difficulty of problems by referring to Shannon's information content contained in constraint-relations. We will compare our methods with sGA by referring to this index.

Related works are as follows: Coevolutionary approach have been studied by many researchers [7], [8]. Especially, coevolutionary approach for solving Constraint Satisfaction Problems is proposed by Paredis [2], [9]. He used two populations which has an inverse fitness interaction, more precisely, the predator-prey relationship, between these populations. Also, schemata-oriented search methods in evolutionary computation have been adopted in several problem solving methods such as Cultural Algorithms and Stochastic Schemata Exploiter [10], [11]. In Cultural Algorithm, usual GA model is associated with a belief space which is similar to the schema space and is used to promote directed evolution of individuals with the beliefs in the GA model. Our GA with invasion algorithm is very similar to Cultural Algorithm in the sense that both methods use additional mechanisms to promote the evolution of usual GA. In our method, however, the information of illegal higher order schemata is utilized to revise the direction of evolutionary process.

2. Constraint Satisfaction Problems

Constraint Satisfaction Problems (CSPs) are a class of problems consisted of variables and constraints on the variables. Especially, a class of the CSPs such that each of the constraints in the problems is related only to two variables are called binary CSPs. In this paper, we treat a class of discrete binary CSPs, where the word discrete means that each variable is associated with a finite set of discrete values (labels) that are candidate values of the variable. An example of the graph coloring problem [15], one of binary CSPs, which is one of the benchmark problems in CSP is delineated in Fig. 1. As depicted in the figure, CSPs are defined by \((U, L, T, R)\), where \(U\), \(L\), \(T\) and \(R\) denote a set of units, a set of labels, unit constraint relations and unit-label constraint relations, respectively. In this 3-coloring problem, i.e., coloring with three colors, \(r\), \(g\) and \(b\), for instance, the set \(U\) of units consists of the nodes in the graph of the given problem. The elements in the set \(L\) of labels denote three colors to be used. The unit constraint relations \(T\) correspond to the edges in the graph of the given problem. The unit-label constraint relation \(R\) is a set of 2-compound labels subject to the constraint relations, where a 2-compound label denote a tuple of labels attached to two variables that are consistent with the unit-label constraint relations. To solve CSPs is to search for solutions such that no constraints are violated, where the graph representation of CSP in Fig. 1(b) called Constraint Network is often used.

We use two indices, tightness and density, for analyzing the difficulties of CSPs [1]. The tightness of an edge \(ij\) is given as the ratio of the number of satisfying 2-compound labels (in unit-label constraint relations) on the edge \(ij\) to the number of all 2-compound labels on the edge \(ij\). Furthermore, the tightness of a problem is given by the average value of tightness of the edges in the problem. The density
of a problem indicates the proportion of constraint relations that actually exist between any pair of nodes. For instance, in Fig. 1, the tightness of the edge $XY$ is calculated as follows: all the constraints on the edge $XY$ are given as $\{(r,g), (r,b), (g,r), (g,b), (b,r), (b,g)\}$, namely, the number of all constraints on the edge $XY$ is equal to 6. Furthermore, the number of 2-compound labels on the edge $XY$ is the same as the product of the number of labels on each node, that is, $3 \times 3$. Hence, the tightness of the edge $XY$ is calculated as $6/9 = 2/3$. Also, the density of the problem is defined as the ratio of the number of the unit constraint relations to the number of all combinations of two nodes among four nodes, namely, $5/\binom{4}{2} = 5/6$.

3. Genetic Algorithm with Layered Population

The proposed method consists of two-level populations: solution-level one and schema-level one. These populations evolve at different levels of abstraction in problem spaces by Genetic Algorithm and affect with each other by two genetic operators: superposition and transcription as depicted in Fig. 2. The genetic information in the solution-level population is used to calculate the fitness value of the schema-level population by superposition operator (see the following subsection). The effective genetic information searched by the schema-level population is transcribed to individuals of the solution-level population. In this paper, the solution-level population and the schema-level population are called H-GA (Host GA) and P-GA (Parasite GA), respectively.

3.1. Superposition Operator

The individuals of the P-GA called P-individuals (P-indiv.'s) represent schemata in the H-GA. Namely, each P-indiv. consists of the alleles used in H-GA and “*” (don't care symbol), which represent a schema in the H-GA. The superposition operator copies the genetic information of a P-indiv. except for “don't care symbol” (“*”), onto one of the individuals of H-GA called H-individuals (H-indiv.'s) in order to calculate the fitness of the P-indiv. Thus, the evolutionary process of this layered population can be regard as a coevolution of individual at different levels of abstraction (cf. Fig. 2).

3.2. Fitness Evaluation of P-GA

P-GA searches for useful schemata in H-GA. Here, the useful schemata in H-GA may be defined as follows: (1) undiscovered useful schemata or simply (2) useful schemata, i.e., those with high average fitness values.

Generally speaking, the fitness value of a schema is calculated as the average fitness value of 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. So, the average value of schema is set to be the average value of "sampled" individuals belonging to the schema. Furthermore, if a 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 where the fitness function of H-individ. 1 and P-individ. 2 are represented as \( f_1 \) and \( F_2 \), respectively.

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

\[
F_2 = \sum_{k=1}^{n} \max(0, f_{i_k} - f_{\tilde{i}_k}), \quad (k = 1, \ldots, n).
\]

Thick lines in Fig. 4 denote the difference between the fitness values of the original H-individ.'s and those of the superposed H-individ.'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.

3.3. Transcription Operator

The transcription operator serves as means of transmitting the effective genetic information searched by the P-GA to the H-GA. This operator propagate genetic information in P-GA to H-Individ. by probabilistically replacing the original H-individ. \( i_k \) with superposed H-individ. \( \tilde{i}_k \). The probability of applying the transcription operation is given as

\[
P_{\text{transcription}} = \begin{cases} 
P_{\text{parasite}} \frac{f_k}{f_{\text{max}} - f_{\text{min}}}, & f_k > 0 \\
0 & \text{otherwise}
\end{cases}
\]

"Australian Journal of Intelligent Information Processing Systems"
where \( f_{\text{max}} \) and \( f_{\text{min}} \) denote the maximum fitness value and the minimum fitness value in the H-GA, respectively, and hence \( P_{\text{parasite}} \) is set to be constant such that \( 0 < P_{\text{parasite}} < 1 \).

4. Genetic Algorithm with Invasion Algorithm

4.1. Framework

We now introduce another evolutionary approach with a layered population inspired by traditional approaches for solving CSPs. We incorporate the Invasion algorithm to GA which is one of the traditional and commonly used methods to solve CSPs, and can find all satisfiable solutions in given problems. First, we will briefly introduce the original invasion algorithm proposed by Seidel [16], which is designed for solving binary CSP. As delineated in Fig. 5, the invasion algorithm keeps the consistency of partial solutions related to the labeling attached to the nodes facing frontier of the invasion process called "front". At each step, the consistent pair of labels is listed up. And, at the final step of the algorithm, all the solutions can be found out by merging the consistent pairs facing at the fronts discovered by the invasion process. The procedure of propagating the front set can be carried out as follows: At each step an uninvaded node adjacent to a node facing the front set is included into the front set. Further, all nodes having links to the external and uninvaded nodes are removed for the front set. In this algorithm, it is very important to devise heuristics to select a node to be included to the front set at each step of the algorithm, since total computational cost is extremely affected by this decision. The Minimum Band-Width Ordering is a well-known such heuristics [1].

We will adopt the basic idea of this invasion algorithm to detect potential illegal tuples of labels. That is, the procedure of our pseudo-invasion algorithm acts as means of building up higher-order partial solutions from lower-order ones by referring to the topology in the graph representation of given problems. In our proposed method, as delineated in Fig. 6, ordinal Genetic Algorithm acts as the CSP solver based on population-based search. At each generation, genetic information which is discovered by the GA population is propagated to constraint network in the upper layer subject to the pseudo-invasion algorithm. Next, pseudo-invasion algorithm is carried out to the genetic information in the constraint network. The objective of the pseudo-invasion algorithm is to build up higher-order partial solutions from the genetic information and to examine whether the genetic information which is satisfiable for some individual will be included in a final satisfiable solution. The results of this examination are
reflected in the GA population in the lower layer in order to promote its evolution. In the following subsections, further details on the extraction of satisfiable genetic information and consistency check in the constraint network are discussed.

4.2. The Extraction of Satisfiable Genetic Information from GA Population

The aim of extraction of satisfiable genetic information from population is to grasp the pattern of genetic information. As delineated in Fig. 7, each of the individuals in population is associated with an error matrix. Each numerals in the error matrix denotes the number of violated constraints at each gene locus. Note that this error matrix can simultaneously be calculated because the fitness function used in this paper is based on the total sum of the numerals in the error matrix. In each individual, the genetic information such that numeral in the error matrix at the corresponding gene locus is equal to zero, is propagated to the constraint network in the figure. In the constraint network, there are many labels without violation against individuals.

4.3. Consistency Check by the Pseudo-Invasion Algorithm

The task of pseudo-invasion algorithm is to probe the minimal illegal genetic patterns, where a minimal illegal genetic pattern means a set of alleles which as a whole do not satisfy the constraints but any subsets of which (and hence consisting of fewer alleles) satisfies the constraints. By searching for the minimal illegal patterns and then by exchanging the genetic information between these layers effectively, it is expected to be able to avoid premature convergence of GA.

The process of consistency check in the pseudo-invasion algorithm is carried out as follows: first, by using the a similar method of invasion algorithm, the front set is progressed. Next, at the front set at each time, the examination whether all the constraints in the front set are satisfied or not is carried out. Hence, we find out the minimal illegal genetic pattern as alleles in the front set, provided that the violation of constraints is found in any pair of alleles in the front set. Furthermore, if such minimal illegal genetic pattern is found in the constraint network, the genetic information in the population which is the same as illegal pattern is looked up, moreover the other alleles are transcripted into the same gene locus.

Also, it is well known that original invasion algorithm necessitates a large amount of computational effort to find all the satisfiable solutions, when the density of constraint network is high. Hence, we have to reduce the maximum size of the front set in order to decrease the number of this consistency check.

5. Experimental Results

In this section, several experimental results in solving CSPs are examined. First, to compare the proposed methods with sGA involving elitist strategy, we carried out computer simulations on General CSPs for a variety of tightness and density. The General CSPs with various values of the difficulty indices, tightness and density, are generated as follows: First, specify the tightness and density. Next, for all combination of two indices and unit constraint relations, decide whether the unit constraint relation is set to each of the pairs of variables or not by taking account of the value of the density. Finally, for all unit constraint relations, the number of unit-label constraint relations is set to be directly proportional
to the tightness. In this paper, the tightness and the density are varied from 10 to 90. The General CSPs used as test beds are set such that both of the number of variables and the size of domain are set to be 10. The fitness function of sGA, that of H-GA and that of GA with Invasion algorithm are set to be $1/(1 + \text{the number of violated constraints})$. In all the experiments, the following GA parameters are used: The population size of all GAs, i.e., sGA, H-GA, P-GA, and GA with Invasion, is set to be 50. The GA parameters for the SGA, H-GA and GA with invasion are set to be of the same value, i.e., the probability $P_c$ of crossover, the probability $P_m$ of mutation and the number of the elitist are set to be 0.8, 0.01 and 5, respectively. Those for the P-GA are set to be $P_c = 0.8$ and $P_m = 0.05$. The elitist strategy is not used in the P-GA. The parameters only used by P-GA are set to be $P_{\text{parasite}} = 0.8$. The number of superposing per one fitness evaluation of P-indiv. is set to be 3. Further, don’t care symbol in the P-GA is generated with a rather high probability, and hence, each P-indiv. initially have few gene loci having their values specified. The specific parameter for GA with Invasion is the maximum size of front set and is set to be 3. Also, the number of runs for each of the tuple ($\text{tightness, density}$) is set to be 100.

Fig. 8 shows the experimental results on General CSPs, where the combination of tightness and density is varied from 10 to 90 with step size 10. The graphs in the lower column in the figure delineate success ratio over 100 runs for each case of tightness and density. The ones in the upper column show the average values of the number (times) of fitness evaluations that were necessary to get satisfiable solutions in success examinations. The left row, the center row and the right row are the results for sGA with elitist (SGA), those for Coevolutionary GA (CGA), and GA with Invasion algorithm (GAwI) respectively. The number of fitness evaluations for the case of CGA is less than the case of sGA for each tuple ($\text{tightness, density}$). Furthermore, in the success ratio (lower column), the region where the success ratio is 1 meaning that all the 100 examinations are perfectly solved, in the case of CGA is broader than the case of SGA. The number of fitness evaluations is the least in the case of GAwI among the three algorithms. However, in the area around ($\text{tightness, density}$) = (10, 80), GAwI cannot effectively solve General CSPs view of success ratio.

Next, the graph coloring problem [15], one of the benchmark problems in CSPs, is examined. The n-coloring problem is a problem to search for the coloring of a graph with n colors. That is, each node should be assigned with a color from among the prescribed n colors such that any adjacent vertices do not share the same color. A method of generating problem instances at least one fully satisfying solution is described in [15]. However, we adopted a naive and simple method which, however, does not assure the existence of satisfiable solutions.

Fig. 9 shows the experimental results on 3 coloring problem with 20 nodes (variables). The GA parameters used in these experiments are set to be the same as those used in the experiments of General CSPs. The tightness for n-coloring problems is constant by definition, namely, $(n - 1)/n$. The horizontal axis in the figure indicates the density. Hence, we investigate the effect of density by examining the cases where $\text{density} = 15, 18, 20, 21, 22, 23$ and 24. The vertical axis in the upper graph denotes the average value of the number (times) of fitness evaluations over the success examinations. The vertical axis in the lower graph denotes the ratio of the success examinations over 100 runs. In the left graph where density is equal to 22, it seems that SGA outperforms CGA and GAwI. In the cases where the density is between 16 and 21, however, the success ratio, i.e., the average number of success trials over 100 trials reveals that SGA will quickly find out satisfiable solutions with, however, low probability because of the simple mechanism of SGA.

Finally, the progress of the evolution process is examined by using information content among unit constraint relations. Namely, to introduce a difficulty index of satisfiability of partial solutions, i.e., schemata in GAs, the information content in the constraint between variables $i$ and $j$ is introduced and is defined as follows:

$$I_{ij} = -\log \frac{\text{the number of 2-compound constraints on variables } i \text{ and } j}{\text{the number of all 2-compound labels on variables } i \text{ and } j},$$

where 2-compound constraints on variables denote the constraints associated with 2-compound labels, i.e., the constraints represented in tuples of labels such that they share a unit constraint relation. If the number of 2-compound constraints is equal to 0, then the information content is regarded to be infinity.
Fig. 8. Experimental results on General CSP: the number of fitness evaluations (UPPER ROW), the ratio of success examination (LOWER ROW), sGA with elitist (LEFT COLUMN), Coevolutionary GA (CGA, CENTER COLUMN), GA with Invasion (GAWI, RIGHT COLUMN)

Fig. 9. Experimental results on the Graph Coloring Problems: the average value of the number of fitness evaluations over the success examinations (LEFT) and success ratio (RIGHT)

Such problems, however, have no satisfiable solutions, that is, we cannot solve such problems. Also, if any two variables have no constraint relations, namely, the number of satisfied 2-compound labels in any pair of variables is equal to the number of 2-compound label in the variables, then the information content is equal to 0. That is, even if a partial solution satisfying only such 2-compound constraint exists, it is of no value to search for such partial solution because any partial solutions can satisfy such 2-compound constraint. Furthermore, we introduce index $D$ of the difficulty of satisfiability of a partial solution $s$ as follows:

$$D_s = \sum_{i,j \in N_s, i \neq j} i_{ij},$$

where $N_s$ denote the set of variables belonging to the partial solution $s$.

Fig. 10 shows the progress of schema formation for each of SGA and CGA on a General CSP with 15 variables. The axis of the difficulty of schemata in the figure exploits the above index, that is, schemata that are more difficult to be satisfied are located on the left part of the axis. The axis of the order denote the order of schemata. We only examined the cases where the order of schemata is 2, 3 or 4. The axis of generation means the generation of population in GA. The hatched area show that satisfiable schemata are dominant in the population. In the figure, SGA fails to find lower order and difficult schemata. Furthermore, SGA cannot search effectively for satisfiable schemata with higher order. On
the other hand, CGA can effectively discover satisfiable schemata at any level of difficulty and of order of schemata. It seems that P-GA works well through combining lower order schemata to make up higher order ones. Finally, it can be readily seen that at any order of schemata, it is different for GAwI to find out effective schemata in the middle difficult range. Further detailed examination shows, however, that GAwI could find out a satisfiable solution at the 28th generation. Also, it turned out that all the satisfiable partial solutions were found out by GAwI. That is, in that middle range, satisfiable solutions cannot be dominant in the population in principle. It seems that GAwI yields rich diversity of population in that range. It is very important for GAs to have such diversity for solving CSPs. It should be noted that we need not such rich diversity in the difficult range so much. For, in the case of the very difficult range, if a satisfiable partial solution is found, it will certainly be a part of a satisfiable full solution. On the other hand, in the case of very easy range, it will hardly be the case that GAwI yields illegal patterns from satisfiable partial solutions.

6. Conclusion

In this paper, two coevolutionary GAs with layered two populations which interact with each other is proposed. The first coevolutionary GA involves two new genetic operators for genetic information exchange between the layers which results in effective exploration and exploitation in the populations improving the search ability of GAs. The second coevolutionary GA incorporates the Invasion algorithm used in traditional CSP solvers. The invasion algorithm is used to verify the consistency of partial solutions by using schema information searched for by the GA in the lower layer. Once, an illegal genetic pattern is found in the process of verification, it is propagated to the lower layer population. Several computer simulations based on General CSPs and graph coloring problems confirm us the effectiveness of these methods in terms of the number of fitness evaluations and of success ratio. Moreover, we introduced the difficulty index of potential satisfiability of schemata by referring to Shannon’s information content. This index provides us ways of examining the progress of schema formation. As the consequence of this examination, the CGA outperforms SGA in the sense of schemata formation, for high order schemata is made up by some low order schemata, i.e., CGA seems to work well. To confirm this conjecture, we have to analyze the behavior of the GA in the upper layer, that is, P-GA, and that of the superpose operator and that of the transcription operator with reference to the index. Furthermore, in GA with Invasion, satisfiable schemata in the middle difficult range were not dominant in the population formed, because such schemata are affected by the genetic information exchange between the layers. This characteristic in the GA with Invasion is very desirable, because it is very difficult to judge in advance whether a schema in the middle difficult range will be involved in a satisfiable full solution or not.

References