# A Computational Model of a Viewpoint-Forming Process in a Hierarchical Classifier System

**Takahiro Yoshimi**
Department of Precision Engineering
Graduate School of Engineering
The University of Tokyo
4-6-1 Komaba, Meguro-ku,
Tokyo 153-8904, Japan
yoshimi@race.u-tokyo.ac.jp
+81-3-5453-5889

**Toshiharu Taura**
RACE: Research into Artifacts,
Center for Engineering
The University of Tokyo
4-6-1 Komaba, Meguro-ku,
Tokyo 153-8904, Japan
taura@race.u-tokyo.ac.jp
+81-3-5453-5890

## Abstract

In an environment where input information to machine learning (ML) systems using production rules has many "properties" and the amount is huge enough, the authors aim for an ML systems with effective performance in finding solutions. When finding solutions, all of the properties of the input information are not always required. Therefore, the authors assume that a certain mechanism which can select specific properties to be focused on will contribute to this purpose. For the realization and discussion of this mechanism, the authors have focused on the Classifier System (CS) which has more advantages than other ML systems. From the authors' point of view, operation processes in the CS are thought to involve this mechanism. However, the CS also involves such "duality" that both the optimization processes of rules for solution finding and the abstraction processes of input information are in a single process, which may lead to problems. In this paper, the authors propose a computational model in which these two processes are explicitly separated. The key concept of the proposed model is the *Viewpoint-Forming Process* for the purpose of using rules for selecting properties to be focused on. This is separate from the standard rules for finding solutions. A computer system is developed to evaluate the utility of this model. The results acquired by applying the model to an example problem are reported here.

## 1 BACKGROUND AND OBJECTIVE

### 1.1 INTRODUCTION

Due to the ease of description, machine learning (ML) systems which use production rules, such as the expert system, have left remarkable results. One of the reasons for this success is the fact that the applied field is limited and input information is fully sorted out along with possible cases.

However, the amount of input information which is handled by such ML systems has now been increasing. For example, when an ML system is located in the same environment as that of a human, and is expected to function as a substitute for a human cognitive mechanism, the input information to the ML system cannot help having a huge number of "properties" (the unit of meaning in input information). The problem is that, as a consequence of the huge amount of input information, the performance of the ML system is often very inefficient.

In this paper, the authors aim at identifying the cause of this problem, and attempt to find a solution. The authors assume that all of the properties in input information are not always required, for example. the human thinking process appears as if it is not processing all of the properties in the input information. In order to fix these properties adequately, another rule on how to select the properties to be focused on is assumed, what the authors call the "Viewpoint-Forming Rule" (VFR). The authors believe that the VFR is determined according to the ease of finding solutions, as a result of interaction with rules for finding solutions, and is not determined by the top-down approach. For

verification of the validity and efficiency among a number of ML systems, the authors employ the Classifier System (CS) as a method of realization and discussion. Note that 'efficiency' here simply means the speed of finding solution without considering other aspects of ML-system.

### 1.2 CLASSIFIER SYSTEM

The CS is the genetics-based machine learning system designed by Holland (Holland 1975). The CS has a mechanism which can perform operations on a production rule (*classifier*, as shown in Figure 1). CS researchers have had successful results with robust rules in complicated and dynamic environments (e.g., Wilson 1985, Hilliard 1987, and Goldberg 1989).

```
<classifier> ::= <condition>:<action>
if <condition> then <action>
<condition>={0,1,#}^l_condition
<action>={0,1}^l_action
```

Figure 1: Standard Definition of *classifier*

According to the increase in the amount of input information to the CS, the rule `<classifier>` tends to have a longer condition part `<condition>`. Because this brings about the expansion of space where the CS searches for effective rules (= solution space), a technique has been developed for the reduction of the `<condition>` length which uses *tags* for rule design. "Well-established tag-based interactions provides a sound basis for filtering, specialization, and co-operation" (Holland 1995) which are the same interests as in this research. Here "well-established" means that the input information and the `<condition>` is fully case-sorted and put in order in minimum length. However, because well-established *tags* may be unclear and difficult at the stage of rule design, it would not solve the fundamental problem of solution space expansion. Thus, it is common to use a number of properties which are allocated sequentially in both input information and the `<condition>`.

Despite the practical difficulties discussed above, an other aspect of CS utilizes the well-defined symbol `#` to solve the solution space problem.

### 1.3 "DON'T CARE" SYMBOL

While the CS owes its success to the advantage of genetic algorithms (GAs), the existence of the symbol `#` in the rule has also contributed. The symbol `#` is usually called "don't care" and it is a wild card which can be substituted with either 0 or 1. The availability of `#` contributes to the reduction of solution space.

Besides the reduction, `#` also has the advantage of selecting properties to focus on when comparing input information with `<condition>`. Holland (Holland 1995) mentioned this feature `#` as "*useful for looking at some specific positions and ignoring others*". This issue is often discussed in relation to the terms of 'Schemata', 'Specificity', and 'Default Hierarchy' in the context of the CS research (Riolo 1989, Smith 1991, Richards 1998). In a sense, the process using `#` is a kind of abstraction or generalization. This feature partly solves the question of this research. However, from a standpoint of this research, it also has problems.

### 1.4 FOCUS OF THIS RESEARCH

In the classical CS, the properties to be focused on using `#` are fixed as a result of hundreds of genetic operations. However, the way they are fixed is embedded in the evolutionary process and the iteration of operations; thus it will not appear until the solution is found. In other words, the process of finding a solution and the process of selecting properties to be focused on are performed together in the classical CS.

The authors believe this "duality" in a single process may lead to three main disadvantages as follows: 1) the process of selecting properties is not explicitly observed from a point of simulation for knowledge acquisition. 2) There would be the possibility of inefficiency due to the two processes working together. Adding to these disadvantages from the authors' viewpoint, when another information is required in order to select properties, for example, the historical record of a series of output information, 3) the CS does not have a framework which can handle different types of input information to select properties. Of course, a Message-Passing Performance System (Holland 1995) can handle it, but this is implicit described and does not satisfy the authors' focus.

Therefore, in this paper, the authors propose the computational model in which the process of finding a solution and the process for selecting properties are separate. Furthermore, the authors assume the existence of another kind of rule, namely, the Viewpoint-Forming Rule (VFR), apart from the standard rule for finding solutions, what the authors call the Solution-Finding Rule (SFR) in this paper in order to distinguish one from the other. Hereafter, the process using the VFR will be called the *Viewpoint-Forming Process*.

As well as the proposition above, the authors con-

structed a prototype system in order to verify the validity of the computational model of the *Viewpoint-Forming Process*, that is, whether it will result in the same or better efficiency and whether there will be any predominant part compared with that of the classical CS.

The rest of this paper is organized as follows. First, in section 2, the authors explain their basic idea, the concept of the *Viewpoint-Forming Process*, for the purpose of separating the "dual" process in the classical CS, and give a framework of how the authors have attempted to realize it. Section 3 follows with a description of the concrete implementation, as well as an example problem considered in the prototype system. The experimental results are also shown and discussed. Finally, in section 4, the authors present conclusions based on the results of the study, and comment on future work.

# 2 VIEWPOINT-FORMING PROCESS

## 2.1 CONCEPT

Figure 2 shows the concept of the *Viewpoint-Forming Process* in production rules and a simple example of processing. In the figure, 6 properties exist in the SFR's condition part and input information. When the ML system faces scene A, it examines and compares only the second and fourth properties, and while facing scene B, only the first and third. Upon facing some special scene C, the ML system examines and compares all properties. Here the VFR which was assumed previously serves to select properties, in order to form a *viewpoint*.

For the utilization of mechanisms using the VFR, the hierarchical rule system has been chosen. The basic concept is that the SFR and VFR sets are located hierarchically, as illustrated in Figure 3. For the availability of different types of input information to each process, this framework provides room for using other kinds of information for the VFR.

To implement this framework, the authors used the Hierarchical Classifier System (HCS), which is explained next.

## 2.2 HIERARCHICAL CLASSIFIER SYSTEM

The general type of CS architecture is distinguished by two policies. One is the Michigan approach (CS-1, Holland 1978) which contains the Apportionment

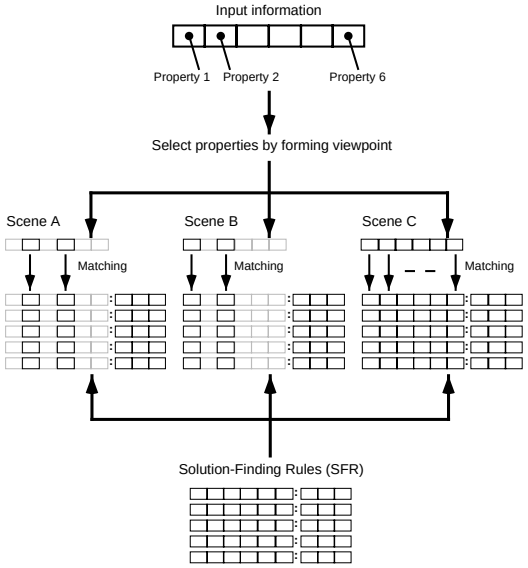


Figure 2: *Viewpoint-Forming Process* in Production Rule

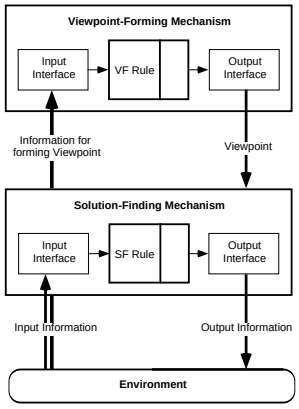


Figure 3: Basic Framework of Proposed Model

Of Credit (AOC) algorithm, i.e., the Bucket Brigade (BB) system, by means of which it evaluates every rule as an "individual". The other is the Pitt approach (LS-1, Smith 1980) which evaluates the rule sets as an individual without using the AOC system.

In this research, the authors aim to observe the *Viewpoint-Forming Process* directly and explicitly in accordance with the number of properties to be focused on in order to examine the efficiency of the VFR. Although the Michigan approach is very common and is powerful in on-line operation, it has the possibility of obscuring the proposed process within the BB system and obfuscating its results because of the sensitive AOC algorithm. On the other hand, the Pitt approach can avoid this and is also relatively easy in that it allows multiple rule sets to be located hierar-

chically. Thus, the authors used this approach when developing the HCS.

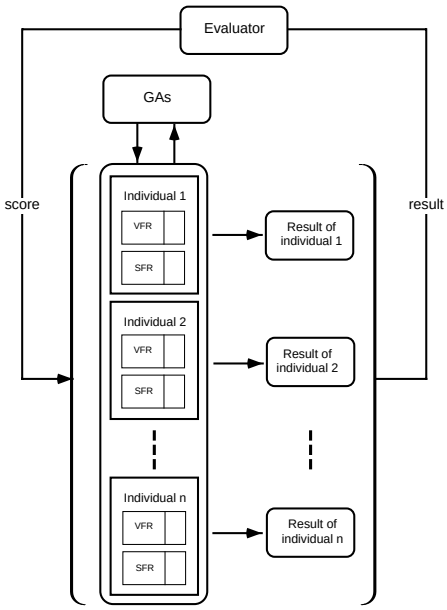


Figure 4: Conceptual Overview of System

Figure 4 shows the conceptual overview of the developed HCS. A single individual consists of two rule sets which are the SFR and VFR. Several individuals are prepared, and each individual is evaluated according to the result that it generates. GAs generate new rules by means of the genetic operations of crossover and mutation, according to the evaluation score.

### 2.3 MECHANISM

The standard definition of a classifier which was previously shown in Figure 1. In this research, depending on the type of rule, there exists a subtle difference between classifiers in an individual.

For the framework shown in Figure 3, in classifiers for the SFR, `<condition>` is the criterion with encoded input information (ordinarily called `<message>` in CS and hereafter) and `<action>` contains the output information. In classifiers for the VFR, `<condition>` is the criterion with `<message>`, same as in classifiers for the SFR, however `<action>` contains the "how to look at" `<message>` information.

Hence, the flow of the transformation of `<message>`, and the matching process is summarized as follows (corresponding to Figure 5).

1. Compare input `<message>` with `<condition>` of a classifier for the VFR; if they match, output `<action>` for the VFR.
2. Select the properties to be focused on within `<message>` according to the `<action>` for the VFR.
3. Compare `<message>` with `<condition>` of a classifier for the SFR partially according to the result of 2; if they match, output `<action>` for the SFR.

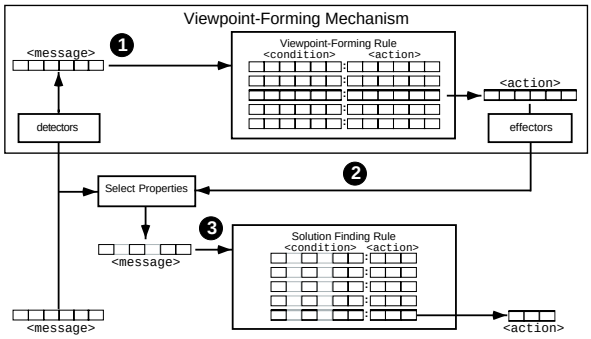


Figure 5: Transformation of `<message>` and Matching Process

If different types of input information are used for VFR, `<message>` in 1 and 2 will be different from that of 3. The type of input information compared with the VFR can easily be changed because the two rules and processes are separate.

## 3 COMPUTATIONAL REALIZATION

For the implementation, the authors employed the classical path-finding problem as an example, since it is easy to understand the behavior and results of the problem.

Besides the examination of the model using standard input information (called "Scene" hereafter) to the VFR, the authors also used, for example, historical record of a series of output information of the SFR (called "History" hereafter) as different types of input information to the VFR.

### 3.1 EXAMPLE PROBLEM

First, the author must explain the setting of the example problem. Agents the number of which is set by parameter are assumed in the simulation space which spans $40 \times 20$ grids. Seven types of "Buildings" and six types of "Marks" exist in the space (Figure 6). Each grid can contain a building and a mark. Some grids contain only a building and some only a mark. Empty grids also exist in the space where the agent can move.

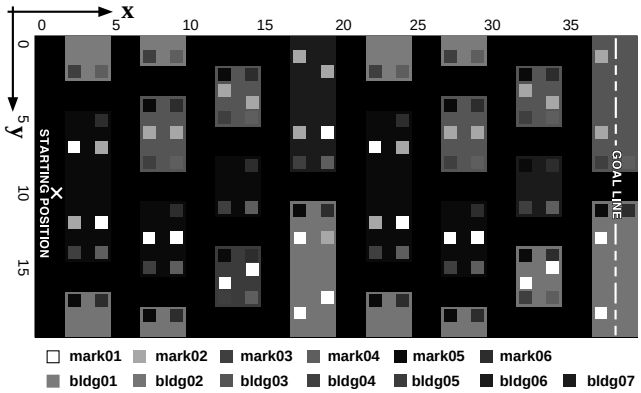


Figure 6: Simulation Space

The starting point is given as a geometric position, and the goal of the agents is given as $x$ coordinate in the simulation space. Then, agents begin finding paths to the goal.

### 3.2 ARCHITECTURE

An overall functional diagram of the process for finding the path to the goal is summarized as follows, corresponding to Figure 7.

1. Input the geometric position where agents start finding paths to the goal; input the $x$ coordinate of the goal.
2. Randomly generate a number of individuals which correspond to each agent and consist of the SFR and VFR classifiers based on the parameters of the "don't care" ratio.
3. Send the information around the agent (Scene) or historical record of agent movement (History) to detectors; translate into `<message>` which is explained in the following point.
4. Compare input `<message>` with `<condition>` of the classifier for the VFR; if match, output `<action>` for the VFR.
5. Select the properties to be focused on within `<message>` according to the `<action>` for the VFR.
6. Compare `<message>` with `<condition>` of the classifier for the SFR partially according to the result of 5; if they match, output `<action>` for the SFR.
7. Send `<action>` to effectors; translate into direction of agent movement.

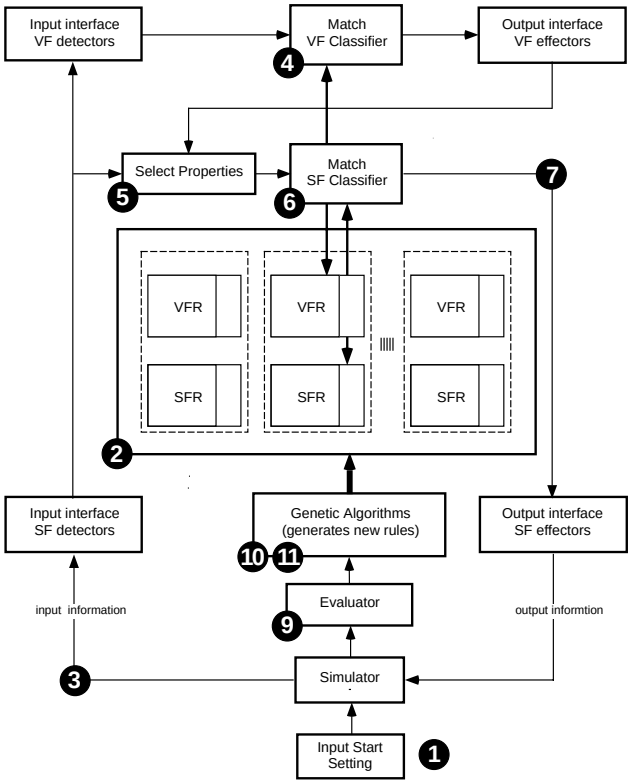


Figure 7: Diagram of Prototype System

8. Repeat 3.∼7. until the agent reaches the goal or faces deadlock.
9. Evaluate individuals according to their fitness function and score them.
10. Select the individuals to approach the goal according to the score.
11. Crossover and mutate selected individuals using genetic operations.
12. Repeat 3.∼11. until at least one agent finds the path and reaches the goal.

### 3.3 GENETIC ALGORITHMS

#### 3.3.1 Coding

An agent can see the grid around itself like Wilson's animat (Wilson 1985). The detector translates the input information around the agent into `<message>`. Also, an agent determines the direction in which to move using the output information decoded by effectors from `<action>`. Figure 8 illustrates how to encode the input information to `<message>` and how to decode `<action>` in agents movement. In the system,

one property is described by 3 bits for the purpose of convenience.

When the historical record of agent movements is used for `<message>` and `<condition>` for the VFR, it is based on a method of encoding that is the reverse of the decoding `<action>` mentioned above.

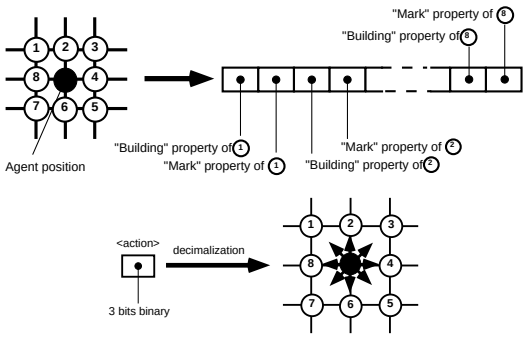


Figure 8: Encoding `<message>` and Decoding `<action>`

### 3.3.2 Fitness Function

For evaluating individuals using two rule sets corresponding to each agent for the selection of GAs, this system uses the simple fitness function

$$Score = \alpha \times x_{displacement} + \beta \times length_{totalpath}.$$

The evaluator calculates the $x$ displacement from the last position of the agent to the goal line and the total length of the agent's paths during each session. The first term contribute to agents advancing in the positive $x$ direction, and the second term to finding the path by wandering about along the $y$ direction. As $\alpha \gg \beta$, the agent which can find the path to progress at least one step in the positive $x$ direction would receive higher score than another agent who only wanders along the $y$ direction, Thus, the fitness function above determines the score of individuals contains two rule sets.

### 3.3.3 Generation-Shift Models

The authors adopt the policy of leaving of plenty of offspring and the inheritance of parental features by the next generation in the GAs, as shown in Figure 9. The authors allow the prototype system to form various VFR sets corresponding to the SFR, which the authors expect will introduce the possibility of searching for better combinations of the SFR and VFR sets. The detail of generation shift process with GAs is as follows.

1. Select certain number of individuals as parents according to the evaluated score.

2. Generate children using crossover and mutation. Here the numbers of crossover probability for SFR and VFR are not 100 % in order to make various type of combinations (ex. not-crossover VFR - crossover SFR).

3. Select mortal individuals according to the reciprocal number of their score and by the use of a simple roulette model. Delete them and be substituted by newly generated children.

This generation-shift model has significance in that it handles the interaction between the SFR and VFR, which performs a kind of feedback to form a better *viewpoint* and allowing for faster solution finding. And this model relates to researches about co-evolution in a sense.

For purposes of comparison, when the authors do not use the VFR, the policy to leave offspring is as shown in Figure 10. There, the number of children is the same as that in the case of using the VFR. In order to make conditions of the experiments more similar, the site is different in each crossover to generate the offspring, assuming that a number of children are generated from parents with good fitness function results.

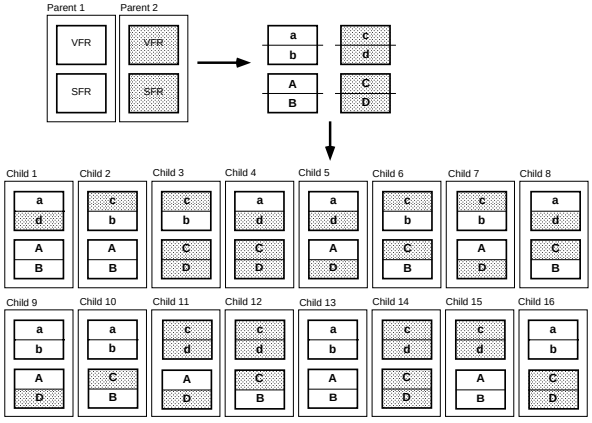


Figure 9: Shift Policy for the VFR

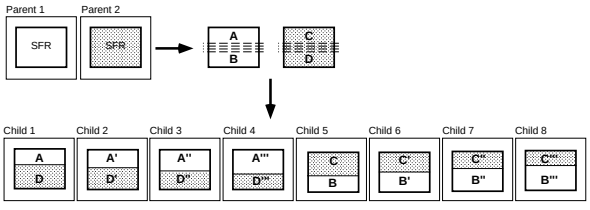


Figure 10: Shift Policy for Comparison with only the SFR

The generation-shift model used in this research works as if it generated new <condition>-<action> pairs. While standard systems (based on the crossover of rule-sets) handled the generation only by mutation or weighting of rules for example, the system in this paper would adapt to various environment even if the number of rules is rather small in an individual. The authors think this brought better results shown in next section than classical CS.

## 3.4 EXPERIMENTAL RESULTS

Here the authors compare three cases with using the VFR (Scene), the VFR (History), and the SFR without the VFR from the following viewpoints. 1) The utility of separating two processes in classical CS, which corresponds to the case using the VFR (Scene). Here, <condition> for the VFR contains "don't care", while <condition> of the SFR does not. The usage of "don't care" is reducing of the solution space and comparison with other cases. Also, <action> for the VFR will contribute to focusing on properties. 2) The availability of different types of input information, which corresponds to the case using the VFR (History). Here the past 5 steps of agent movements are the input information and <condition> for the VFR. The <condition> contains "don't care" and the usage is the same as 1). <action> is also the same as 1).

Because it is important to select the properties to be focused on, in the two cases above, the ratio for focusing on adequate properties is lower than the ratio which is left by "don't care" in <condition> for the VFR. Here the authors use the VFR as if it compensates for the disadvantage of the standard mechanism using "don't care".

3) The standard process of the CS corresponds to the case with only the SFR without the VFR. This is for comparison with 1) and 2). The <condition> for the SFR contains the standard "don't care".

In these cases, the authors employ several "don't care" ratios in <condition> for the purpose of observing the efficiency of the VFR, even though the ratio changed.

### 3.4.1 Experimental Parameters

Table 1 shows the common parameters in all experiments. Table 2 shows the experimental parameters for each case, that is, using the VFR (Scene) with the SFR, using the VFR (History) with the SFR, and using only the SFR .

### 3.4.2 Results and Discussion

As an example result, Figure 11 shows the path acquired in the trial with a "don't care" ratio of 80 % in the VFR <condition> without using "don't care" in the SFR <condition>.

Table 1: Common Parameters in Experiments

| **Common items** | **value** |
|---|---|
| Number of rules / individual | SFR:300 VFR:300 |
| Number of individuals | 60 |
| Number of parents | 8 |
| Number of children | 48 |
| Value $\alpha$ in Fitness Function | 10 |
| Value $\beta$ in Fitness Function | 0.001 |
| Start position | (1,10) |
| Goal line | x=38 |

Table 2: GA Parameters for Experiments

upper : using the VFR (Scene) with the SFR
middle : using the VFR (History) with the SFR
lower : using only the SFR

| **item** | **SFR** | **VFR** |
|---|---|---|
| Crossover probability | 0.5 | 0.5 |
| Mutation probability | 0.01 | 0.01 |
| <condition> length | 48 | 48 |
| <action> length | 3 | 16 |
| # ratio in <condition> | 0 | 65,70,75 80,85 |

| **item** | **SFR** | **VFR** |
|---|---|---|
| Crossover probability | 0.5 | 0.5 |
| Mutation probability | 0.01 | 0.01 |
| <condition> length | 48 | 15 |
| <action> length | 3 | 16 |
| # ratio in <condition> | 0 | 65,70,75 80,85 |

| **item** | **SFR** |
|---|---|
| Crossover probability | 1.0 |
| Mutation probability | 0.01 |
| <condition> length | 48 |
| <action> length | 3 |
| # ratio in <condition> | 65,70,75,80,85 |

Figure 12 shows the overall results for the all trials. It shows the averages for the three trials until the 500th generation, each of which uses a different random seed in the initial generation of individuals. Although they cannot be directly compared because the three cases involve different conditions, such as the location of "don't care" and length of <condition> (which means that the solution space of VFR (History) was narrower

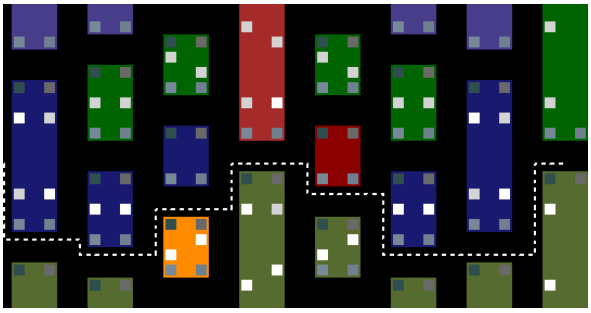

Figure 11: Example of Acquired Path (using the VFR, "don't care" 80%)

than others), here the authors can see the efficiency of mechanisms using the VFR compared with the standard CS despite the ratio of "don't care" changes.

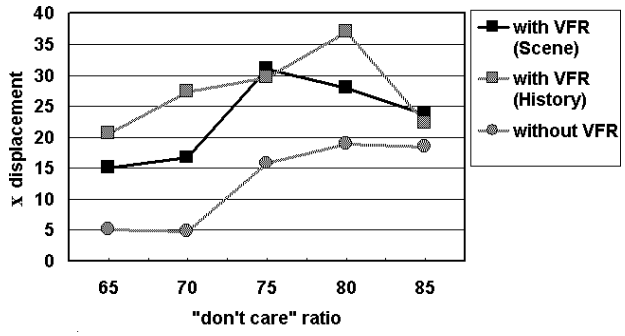


Figure 12: Overall Results of Example Problem

Figures 13∼15 show the three sections of Figure 12 at "don't care" ratio of 70 %, 75 %, and 80 % which are the transitions until the 500th generation. They also show the efficiency of mechanisms using the VFR which is the realization of the *Viewpoint-Forming Process* proposed here.

By separating the process of selecting properties to be focused on, which the authors call the *Viewpoint-Forming Process*, from the process of finding solutions and realizing them using HCS, the former process can be explicitly observed here.

The results shown above confirm that the separation of the two processes contributed to the increase in efficiency.

By representing the results using a historical record of a series of output information in the process of selecting properties, the authors developed a framework which can handle different types of input information.

With the proposition of the *Viewpoint-Forming Process* concept using the VFR, the construction of a computer system based on the model and the good results obtained by applying the model to an example problem, the authors confirmed that the system based on this proposal gives the same or better performance as using the classical CS, and verified the validity of the computational model proposed here.

Considering these good result, the authors also believe that this framework has good potential for practical use and scientific analysis.

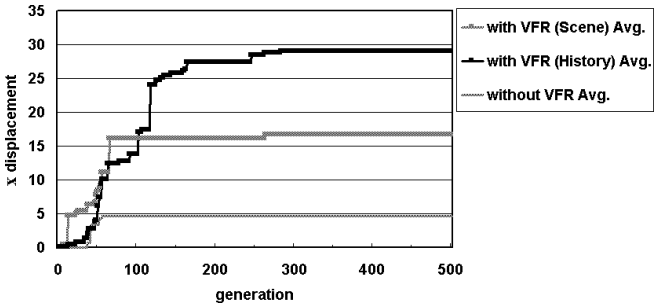


Figure 13: Results of "don't care" 70%

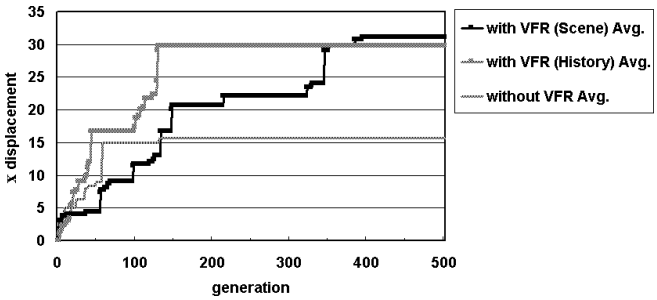


Figure 14: Results of "don't care" 75%

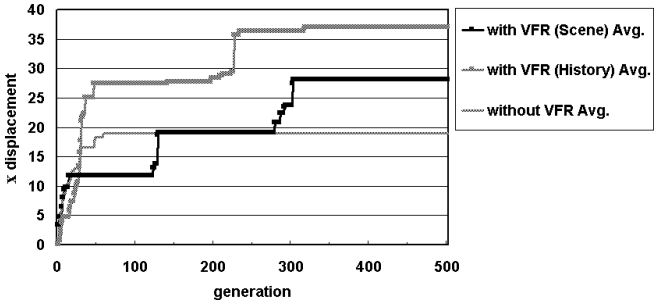


Figure 15: Results of "don't care" 80%

# 4 CONCLUSIONS AND FUTURE WORK

## 4.1 CONCLUSION

Towards the problem of effective performance in finding solutions in the environment where input information to the ML system using production rules has many properties and the amount is huge enough, the

authors put forth an answer by means of proposing the existence of a rule for the *Viewpoint-Forming Process*.

### 4.2 FUTURE WORK

In future work, in computational simulation, the authors are attempting to carry out asynchronous genetic operations of the SFR and VFR which will clarify the interactivity between them. This issue relates "co-evolution" to the context of GA research. Other work involves an application to practical problems such as the simulation of the thinking process in design, in which the potential of mechanisms using the VFR are utilized. Also the authors are interested in the construction of a computer-aided engineering system based on the computational model proposed here.

### References


J.H.Holland (1975), Adaptation in Natural and Artificial Systems, MIT Press.

S.W.Wilson (1985), Knowledge Growth in an Artificial Animal, Proceedings of an International Conference on Genetic Algorithms and Their Applications, 16-23, Morgan Kaufmann.

M.R.Hilliard, G.E.Liepins, M.Palmer, M.Morrow, and Richardson, J. (1987), A Classifier-Based System for Discovering Scheduling Heuristics, Proceedings of the Second International Conference on Genetic Algorithms, 231-235, Morgan Kaufmann.

D.E.Goldberg (1989), Genetic Algorithm in Search, Optimization, and Machine Learning, MIT Press

J.H.Holland (1995), Hidden Order, Addison-Wesley.

R.L.Riolo (1989), The Emergence of Default Hierarchies in Learning Classifier System, Proceedings of the Third International Conference on Genetic Algorithms, 322-327, Morgan Kaufmann

R.E.Smith (1991), Default Hierarchy Formation and Memory Exploitation in Learning Classifier System, Technical Reports #91003, The Clearinghouse for Genetic Algorithms, University of Alabama.

R.A.Richard (1998), Classifier System Metrics: Graphical Depictions, Proceedings of the Third Annual Genetic Programming Conference, 652-657, Morgan Kaufmann

S.F.Smith (1980), A Learning System Based on Genetic Algorithm, University of Pittsburgh, Ph.D. dissertation.