The Third AFSS(1998). 489－494

A Design of Multi-Field User Interface for Simulated Breeding

Tatsuo UNEMI

Dept. of Information Systems Science, Soka University
1-236 Tangi-cho, Hachiôji, Tokyo 192-8577 Japan
TEL: +81-426-91-9429, FAX: +81-426-91-9312, Email: unemi@iss.soka.ac.jp
URL: http://www.intlab.soka.ac.jp/~unemi/

Abstract: This paper describes a design of graphical user interface for a simulated breeding tool with *multi-field*. The term *field* is used here as a population of visualized individuals that are candidates of selection. Multi-field interface enables the user to breed his/her favorite phenotypes by selection independently in each field, and he/she can copy arbitrary individual into another field. As known on genetic algorithms, a small population likely leads to premature convergence trapped by a local optimum, and migration among plural populations is useful to escape from local optimum. The multi-field user interface provides easy implementation of migration and wider diversity. We show the usefulness of multi-field user interface through an example of a breeding system of 2D CG images.

Keywords: simulated breeding, interactive evolutionary computing, graphical user interface, genetic art.

1 Introduction

The method *interactive evolutionary computing* that let the user select his/her favorite individuals among the population is going to be applied to many fields as a useful method to optimization following subjective preference. The key points to make the application successful are not only on design of gene coding as construction of solution space, but also on design of efficient and effective usr interface to decrease the user's workload.

In this paper, we propose a design of graphical user interface using *multi-field* for simulated breeding method. The term *field* is used here as a population of visualized individuals that are candidates of selection. Multi-field interface enables the user to breed his/her favorite phenotypes by selection independently in each field, and he/she can copy arbitrary individual into another field. As known on genetic algorithms, a small population likely leads to premature convergence trapped by a local optimum, and migration among plural populations is useful to escape from local optimum. The multi-field user interface provides easy implementation of migration and wider diversity.

In the following sections, we consider some aspects of evolutionary process on optimization and diversity, propose a design of multi-field GUI, and describe some remarks through a breeding tool of two dimensional computer graphics (2D CG) image as an example system.

2 Optimization and diversity

Billions years of evolution has produced very wide variety of sophisticated natural organisms as we know as in intelligent information processing by brain, self-repairment of born and skin, physiological chemical processing of metabolism, and so on. We can see this process as optimization of self-reproductivity in terms of efficiency and robustness against environmental changes. It is a reasonable idea to introduce the similar process of evolution into engineering as a optimization method. Techniques called evolutionary computation can provide powerful tools to find a feasibly optimal solution for some complex structures though they may waste huge amount of computation resources.

We likely suffer a complicated multi-modal landscape in a structural optimization problem because the search space constructed by solution candidates is usually high dimensional. It is necessary to examine as many candidates as possible to find the best solution because each candidate has a lot of neighbors in a high dimensional space. One of the key techniques for successful search is a method to keep diversity of individuals in population. *Island model* [3] is one of the methods to keep diversity similar to the others such as large size of population, *local*

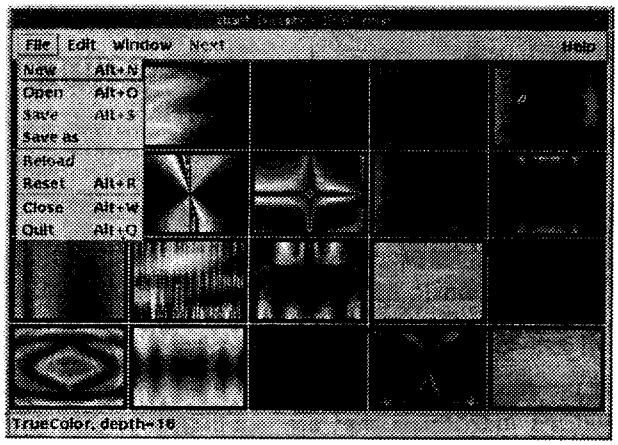

Figure 1: The field window and the file menu. Gene population of each field can be saved into a file. "New" button spawns a new field window that contains a population with randomly generated genes.

selection [1], *minimal generation gap* [2], and so on.

After geographical separation of a continent into islands, a population of a specie that lived in a continent is divided into sub-populations in each island, and they reach usually distinct organisms through independent evolutionary process since mating between different islands is prohibited and there are many sub-optimal points for the structure of an organism. As the scheme theory [4] in the genetic algorithm indicates, a crossover operation, a combination of parts of different genomes, has a possibility to spawn a better individual by combination of good genes from different individuals. We can expect to get better solutions by migration of individuals among different islands after reaching some convergence in each island.

It is impossible to take a large size of population in the framework of interactive evolutionary computing because the user has to observe as all of phenotypes in the population as possible to evaluate them instead of predefined fitness function in the ordinary framework of evolutionary computation. Small size of population often leads the evolutionary process to undesirable direction such as unstability with genetic drift and premature convergence into a local optimum. It is necessary to introduce some method to keep diversity even with a small population. The multi-field user interface proposed below is a method to bring a similar effect with island model into a tool of interactive evolutionary computing.

3 Design of multi-field user interface

We propose a *multi-field user interface* that enables to keep a wide diversity of gene with a small population in a similar manner with the island model. We use a breeding tool of 2D CG images named *sbart2.2b* [5] developed by the author as an example. This tool's basic concept came from combination of Genetic Programming [6] and Simulated Breeding [7], originally proposed by Karl Sims as Artificial Evolution [8].

3.1 Field window

Visualized individual phenotypes of a population are displayed together in a *field window* as shown in Figure 1. We assume that it is possible to visualize each phenotype with not so height computational cost, and that the population size is about from sixteen to thirty. A field window is divided into from 4×4 to 5×6 grids, showing each individual in each rectangle of grid. It should be possible to produce an arbitrary number of field windows by user's command. Genetic operations without migration are performed in each field. Following the simulated breeding method, the user explicitly selects one or more individuals in the field as parents of the next generation. When only one individual is selected then "Next" button is pushed, all of individuals except selected one are replaced by mutants of selected individual. When more than one individuals are selected, all of individuals are replaced by children of selected parents produced with crossover operation.

Figure 1 shows a field window and the file menu of *sbart*. If the user selects "New" button, a new field window appears on the screen initialized with randomly generated genes. A population of genotypes in the field can be saved into a disk file. "Open" button opens an existing file of which name the user selects, and loads its contents as a new population in a new field window. The file menu includes "Save" button that saves a current population, "Reset" button that resets a population with random genes, and so on.

3.2 Migration among fields

Sbart has two types of methods to migrate an individual between fields. The first is to move it by *copy and paste* using the *edit menu* shown in Figure 2 or using *short cut key* corresponding to each edit operation. The operation includes four steps, selecting the indi-

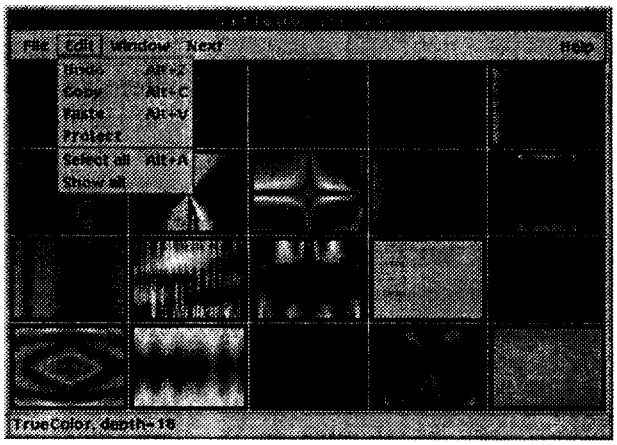

Figure 2: Field window and the edit menu. Genome of each individual can be copied between a common copy buffer and any field grid.

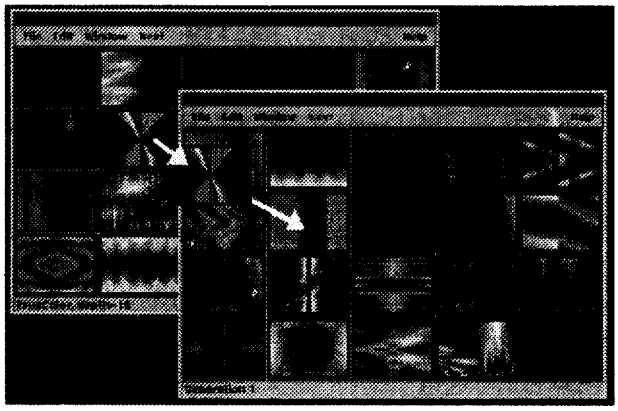

Figure 3: Migration of individual by *drag and drop.*

vidual that the user wish to move on the field first, copying it into the copy buffer, selecting the individual that should be discarded by overwriting, and then pasting it. Selection of an individual on the field is done by pointing it and clicking the left mouse button. A red border frame of a rectangle area indicates the individual is selected. It takes six steps if the user uses the edit menu to both copy and paste because of invoking the edit menu from the menu bar. Short cut keys can reduce it by two steps.

The other is by *drag and drop.* As Figure 3 shows, the user can copy any individual by pointing it with the mouse cursor, pressing the left button, moving the mouse pointer keeping the mouse button pressed, and releasing the button at the destination rectangle. A small window showing the individual image moves following the mouse's move. This method need less number of operation than the first one. It is easier for the user who well know another application with similar operation such as a file manager.

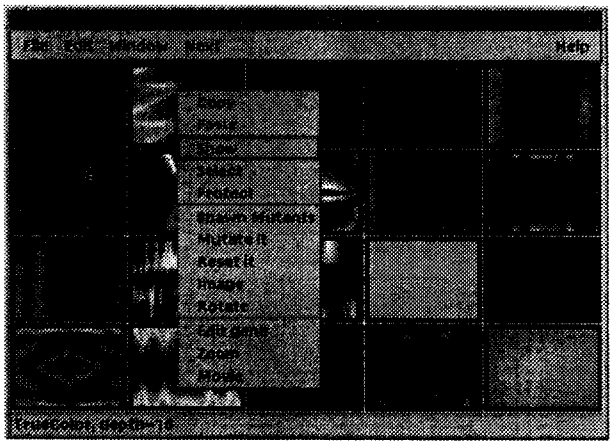

Figure 4: Individual menu is invoked when the user moves the mouse pointer to the displayed individual in a field window and pushes the right button. Operations for pointed individual can be done.

3.3 Protection of individual

Not only to save genes in a disk file, the user sometimes wish to keep some interesting individuals without modification temporally. It is realised by protection of individual in *sbart.* The user can protect his/her selected individual by pushing "Protect" button in the edit menu as shown in Figure 2. Or, he/she also can do it from the *individual menu* invoked by clicking the right mouse button on the displayed individual. The latter method is better in terms of the number of basic operations. This *individual menu* also includes "Copy," "Paste," and other operations for indexed one individual.

The alternative design of individual protection is to facilitate the other type of window that keeps the arbitrary number of genome as a profile. This method may lead to a filing system including library files, however we have not tried to implement it yet. Some method for efficient retrieval for desired genome from large scale data base will be needed for this type of filing system.

4 Effects of multi-filed interface

Populations processed through independent evolution usually reach unique features for each. Fluctuation of subjective evaluation criteria and multi-objectiveness provide source of a wide variety of niches for individuals. Additionally, variable length of chromosome such as in Genetic Programming provides wider diversity because of complexity of the search space.

Figure 5 shows offsprings produced by crossover between individuals that came from different fields.

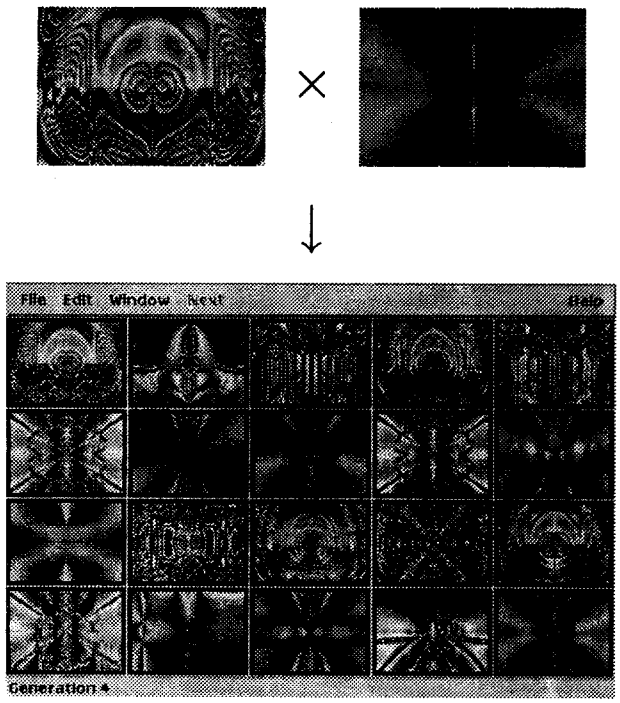

Figure 5: An example of offsprings spawned by migration and crossover between two individuals bred independently in the different fields.

Some part of features of both parents inherit to the children, but it is unknown which feature remains because the crossover points are decided randomly. If the gene coding allows redundant representation or non-effective part, it may produce children of more unexpected features. The concrete information (function) of the genome of parents and children in Figure 5 is shown in Appendix.

5 Conclusion

Introducing multi-field user interface proposed here makes it possible to keep wide diversity of individuals by a small population in an application of interactive evolutionary computing. We described design and effects of this method using *sbart2.2b* as a typical example. It can be applied to another types of domains and frameworks other than simulated breeding. *Sbart2.2b* includes also *genome editor* that manipulate gene directly, just like a genetic operation by a biologist, and domain-dependent operations, such as exchange between X and Y coordinates for 2D CG. These additional functions help to provide more powerful effects for wider diversity through combination with the multi-field interface.

Our future work should include some cognitive evaluation of the user interface through a protocol analysis and statistical analysis of the effects for keeping wide diversity. Some researchers in the field of evolutionary computation recently have interests in theoretical and experimental analysis on landscape and evolutionary process of variable length of chromosome [9]. We may be able to expect some fruitful hints from their results in near future.

The information related to *sbart* is available from the following URL including the source and binary codes.
`http://www.intlab.soka.ac.jp/~unemi/sbart/`

Acknowledgment

The tool for breeding 2D CG images named *sbart* was developed by the author inspired by the idea of Artificial Evolution by Karl Sims [8]. The Sims' system works on a CM-2 massively parallel machine with sixteen SGI graphics workstations. *Sbart* is one of the descendants that works on a Unix workstation with X Window system with original extension including multi-field user interface described here. Since starting distribution of that program on the Internet in 1993, a lot of persons have given us advices and encouragement. We would like to thank all of them, especially Mr. Karl Sims for his kind permission for distribution of the program code.

References

[1] Sarma, J. and De Jong, K. : An Analysis of Local Selection Algorithms in a Spatially Structured Evolutionary Algorithm. in *Proc. of the Seventh ICGA*, 181–187, Morgan Kaufmann (1997).

[2] Satoh, H., Ono, I., and Kobayashi, S. : A New Generation Alternation Model of Genetic Algorithms and Its Assessment, *Journal of Japanese Society of Artificial Intelligence*, Vol. 12, No. 5, 734–744 (1997 in Japanese).

[3] Pettey, C. B., Leuze, M. R., and Grefenstette, J. J. : A Parallel Genetic Algorithm, in *Proc. of the Second ICGA*, 155–161, LEA (1987).

[4] Goldberg, D. E.: Genetic algorithms in search, optimization and machine learning, Addison-Wesley (1989).

[5] Unemi, T. : The World of Arts that Artificial Life Creates, in T. Shibata and T. Fukuda (eds),

Near Future of Artificial Life, Jiji-tsuushin-sha, 69-86, (1994 in Japanese)

[6] Koza, J. R. : Genetic Programming: on The Programming of Computers by Means of Natural Selection, MIT Press (1992).

[7] Dawkins, R.: The Blind Watchmaker, Longman, Essex (1986).

[8] Sims, K.: Artificial Evolution for Computer Graphics, *Computer Graphics,* Vol. 25, No. 4 (1991).

[9] Iba, H.: Recent Research on Evolutionary Computation, *Journal of Information Processing Society of Japan*, Vol. 39, No. 1, pp. 32–36 (1998 in Japanese)

Appendix: Genomes of individuals in Figure 5

Parents at left side

```
pow(hypot(max(YX0,0XY),sin(hypot(0YX,0XY))),hypot(max(sin(hypot(0YX,0YX)),0XY),YX0))/--0.289
```

Parents at right side

```
sqrt(cos(sqrt(XY0))/(sin(hypot(YX0/X0Y,0YX)--0.805)--0.383-mix(-1.125,1.453)))
```

Children

(upper one of first row → lower one of first row → second row → ... → lower one of fifth row)

```
1,1:pow(hypot(max(YX0,0XY),sin(hypot(0YX,0XY))),sin(hypot(YX0/X0Y,0YX)-0.805)--0.383)/-0.289
2,1:sqrt(cos(sqrt(XY0))/(sin(hypot(YX0/X0Y,0YX)-0.805)--0.383-sin(hypot(0YX,0YX))))
3,1:pow(sqrt(XY0),hypot(max(sin(hypot(0YX,0YX)),0XY),YX0))/-0.289
4,1:sqrt(cos(sqrt(XY0))/(sin(hypot(YX0/X0Y,0YX)-0.805)--0.383-0YX))
1,2:pow(hypot(max(YX0,0XY),sin(hypot(0YX,0XY))),hypot(max(sin(hypot(0YX,0YX)),0XY),YX0))/(YX
    0/X0Y)
2,2:sqrt(YX0/(sin(hypot(YX0/X0Y,0YX)-0.805)--0.383-mix(-1.125,1.453)))
3,2:pow(hypot(max(YX0,sin(hypot(YX0/X0Y,0YX)-0.805)--0.383-mix(-1.125,1.453)),sin(hypot(0YX,
    0XY))),hypot(max(sin(hypot(0YX,0YX)),0XY),YX0))/-0.289
4,2:sqrt(cos(sqrt(XY0))/(YX0-mix(-1.125,1.453)))
1,3:pow(hypot(max(YX0,0XY),sin(hypot(YX0/X0Y,0YX)-0.805)),hypot(max(sin(hypot(0YX,0YX)),0XY)
    ,YX0))/-0.289
2,3:sqrt(cos(sqrt(XY0))/(sin(hypot(YX0/X0Y,0YX)-pow(hypot(max(YX0,0XY),sin(hypot(0YX,0XY))),
    hypot(max(sin(hypot(0YX,0YX)),0XY),YX0)))--0.383-mix(-1.125,1.453)))
3,3:pow(hypot(max(YX0,0XY),sin(hypot(0YX,0XY))),hypot(max(sin(hypot(0YX,0YX)),0XY),sin(hypot
    (YX0/X0Y,0YX)-0.805)--0.383-mix(-1.125,1.453)))/-0.289
4,3:sqrt(cos(sqrt(YX0))/(sin(hypot(YX0/X0Y,0YX)-0.805)--0.383-mix(-1.125,1.453)))
1,4:pow(hypot(max(YX0,0XY),sin(hypot(0YX,0XY))),hypot(max(sin(hypot(0YX,sin(hypot(YX0/X0Y,0Y
    X)-0.805)--0.383)),0XY),YX0))/-0.289
2,4:sqrt(cos(sqrt(XY0))/(sin(hypot(YX0/X0Y,0YX)-0.805)--0.383-mix(max(sin(hypot(0YX,0YX)),0X
    Y),1.453)))
3,4:pow(hypot(max(YX0,sin(hypot(YX0/X0Y,0YX)-0.805)--0.383),sin(hypot(0YX,0XY))),hypot(max(s
    in(hypot(0YX,0YX)),0XY),YX0))/-0.289
4,4:sqrt(cos(sqrt(XY0))/(sin(hypot(YX0/X0Y,0YX)-0.805)-YX0-mix(-1.125,1.453)))
1,5:pow(hypot(max(YX0,0XY),sin(hypot(0YX,sin(hypot(YX0/X0Y,0YX)-0.805)--0.383-mix(-1.125,1.4
    53)))),hypot(max(sin(hypot(0YX,0YX)),0XY),YX0))/-0.289
2,5:sqrt(cos(sqrt(hypot(max(YX0,0XY),sin(hypot(0YX,0XY))))))/(sin(hypot(YX0/X0Y,0YX)-0.805)--
    0.383-mix(-1.125,1.453)))
3,5:pow(hypot(max(YX0,0XY),sin(hypot(0YX,0XY))),hypot(max(sin(hypot(0YX,0YX)),0XY),YX0/X0Y))
    /-0.289
4,5:sqrt(cos(sqrt(XY0))/(sin(hypot(YX0/X0Y,0YX)-0.805)--0.383-mix(-1.125,max(YX0,0XY))))
```