Implementing Evolutionary Self-Organizing Maps with the Genetic Operations of Graph Evolution Theory

Maiga Chang and Jia-Sheng Heh
maiga@ms2.hinet.net, jsheh@ice.cycu.edu.tw
Dept. of Info and Comp Eng., Chung Yuan Christian Univ., Chung Li, 320, Taiwan

Abstract **–** This paper analyzes the genetic operations of a new evolution mechanism proposed by us for improving the capability to deal with graph-form solutions in the real world of genetic algorithms based on the theories of GAs and GPs. A prototype of graph evolution with genetic operations is implemented and applied to some graph-related systems with the Irish-student classification data. Evaluation between conventional optimization mechanisms and graph evolution theory is also made for proving the advantage of using graph evolution. Be notable is the graph evolution theory proposed in this paper can cover most applications of GAs and GPs.

I. INTRODUCTION

Conventional optimization techniques are always designed for specific problems, and the results made from these sorts of optimization techniques might be not accurately enough if there are minor changes applied to the current solved problem. [28][18] There are several issues involved for the considerations and limitations of those conventional optimization techniques, such as learning speed and local minimum/maximum. On the contrast, with the characteristics of adapting to environment (adapting to problem) and evolving from generation to generation (finding global optima), evolutionary algorithms are much easier to fit different problem domains. [1] Recently, genetic algorithms provide the most considerable solving mechanism of evolutionary algorithms as applied in the area of industrial engineering. [14] Genetic algorithms (and its derived – genetic programming) translate any possible solution for a problem into a string (either tree-form) chromosome. [17][20]

Unfortunately, not all of solutions in the real world can be formulated as the list chromosomes simply. Many researchers before used some tree-like or mesh-like chromosomes to replace the list chromosomes for solving problems in the real world. [22][9] Although the graph chromosomes are possible to provide more general solutions for problems ([10][25]), there is not a complete and systematic theoretical analysis made for the graph evolution. [15][16][7]

For graph-form solutions, this paper presents a prototype of evolutionary algorithm, which is so-called *Graph Evolution*. Consistent formulations for genetic phase in the evolutionary algorithms are analyzed and designed between graph evolution and genetic algorithm. However, since the paper length is restricted, some of detailed proves are ignored and will be discussed in the complete paper. Finally, this paper tries to apply the prototype of graph evolution to graph-related system such as Kohonen's SOM. [22] Evaluation of the modified systems is made for proving the advantage of using graph evolution instead of traditional problem-solving mechanisms.

Section 2 describes the fundamental of priori-knowledge about the concept of graph theory, which is useful for transforming genetic operations to graphs. Besides, the widely-known types of evolutionary algorithms, genetic algorithm and genetic programming, is also be illustrated in Section 2 for the consistency analyzed and designed in Section 3 and Section 4. Detailed analysis of genetic phase of graph evolution is done in Section 3 and Section 4. Section 3 will focus on the unary operations in the genetic phase such as reproduction and mutation; and the binary operation, crossover will be discussed in Section 4. Experiment systems are implemented and evaluated by Section 5 with the modified algorithms in which our graph evolution is applied to improve performance. Section 6 gives a summary and discusses what can be done in future.

II. GENETIC ALGORITHMS (GA/GP)

Evolutionary algorithm is a stochastic optimization technique derived from simulating the natural evolutionary process originally proposed by Charles Darwin in 1959. [11] Such algorithm can often outperform conventional optimization methods when applied to difficult real-world

0-7803-7898-9/03/$17.00 ©2003 IEEE

problems. [4][27][24] There are currently three main avenues of this research: *genetic algorithms* (GAs), *evolutionary programming* (EP) and *evolution strategies* (ESs). [2] Among them, genetic algorithms are perhaps the most well-known evolutionary algorithms today. [26][5]

Genetic algorithm starts with an initial set of random solutions, called a *population*. Each *individual* in the population is called a *chromosome*, representing a solution. A chromosome is a string of symbols; it is usually, but not necessarily, a binary bit-string. [17] The chromosomes evolve through successive iterations, called *generations.* During each generation, the chromosomes are evaluated, using some measurement of *fitness*. [13]

After all the individuals (*the parents*) of the generation are evaluated by the fitness function, it is possible to produce a new generation formed by new individuals (*the offsprings*). Those fitter chromosomes with high fitness values have higher probabilities of being selected. After several operations these chromosomes can represent the optimal or suboptimal solutions to the problem. In fact, there are only two phases in genetic algorithms: [14]

1. *genetic phase*: reproduction, mutation and crossover,
2. evolution phase: selection.

The *genetic phase* mimics the process of heredity of chromosomes to create new offsprings at each generation, and the *evolution phase* mimics the process of *Darwinian evolution* to create populations from generation to generation. [19] Each operation in both phase can be represented as different operators, such as s operator for mutation, p operator for crossover and z operator for selection. Applying evolution process to the current generation, a set of chromosomes is selected to transform the whole new generation, called the *offsprings*, which will continuously perform several different genetic operations, such as crossover and mutation.

Although the definitions of genetic algorithms give sufficient conditions for researching, some necessary background is still needed for analyzing other kinds of chromosome forms of evolution, such as tree, mesh and even graph. Genetic Programming is a sort of tree form GAs and defines several kinds of crossover operators. [23] Since this paper try to analyze the genetic phase for graph evolution, definitions and principles of GAs and binary operators of the tree-form in GPs should be taken into consideration. [6]

This paper will concentrate to working on the analysis of genetic operations. Before the genetic operations are analyzed, fundamental definitions of *graph evolution* should be illustrated firstly.

Assumed that $P^{(t)}$ is the *population* at the specific generation t and contains several individuals. [3] An *individual* $\vec{a}$ is formed with the *chromosome structure* $\vec{x} \in X^{(t)}$, where $X^{(t)}$ is the *structure space* at generation t. From time to time, this chromosome structure $\vec{x}$ is controlled by some *strategy parameters* $\vec{s} \in S$, where S is the *strategy parameter space*. [12] However, we decide not include *Evolution Strategies* temporary in this moment.

Definition 1. An *individual* $\vec{a}$ with *graph chromosome* is defined as:

$$\vec{a} = \left(\vec{x}; \vec{s}\right) = \left(\tilde{x}_1, \tilde{x}_2; \vec{s}\right),$$

where the chromosome structure is corresponding to $\tilde{x}_1 = V$ (the vertex set) and $\tilde{x}_2 = E$ (the edge set) and $\vec{s}$ is the strategy parameters. □

To follow this definition, an example is given in Figure 1 to demonstrate how a *graph chromosome* can be.

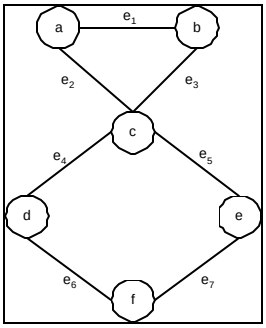


Fig. 1. A graph chromosome

III. UNARY OPERATIONS

Similar to the string chromosomes in genetic algorithm, graph chromosomes also have two kinds of operations for graph evolution: unary operations and binary operations. The unary ones are investigated at first. Let $G(V, E)$ be the graph chromosome before the unary operation of graph evolution, and $G'(V', E')$ is respected as after the operation. In general, a single *unary operation* on such graph chromosome makes only a single increment or decrement of elements in node/edge sets, which falls into one of the following combinations in Table 1. As shown in Table 1, different operations give different rules, which can be used to

determine what kind of a descendant graph will be produced through such an evolution process. Lemma 1 gives more formal descriptions.

TABLE 1. POSSIBLE COMBINATIONS OF MUTATIONS IN GRAPH CHROMOSOME

Simple operations	$E \subset E'$	$E = E'$	$E \supset E'$
$V \subset V'$	Node Diff.	X	X
$V = V'$	Edge Diff.	Reproduction	Edge pruning
$V \supset V'$	X	X	Node pruning

Lemma 1. For the unary operation of a graph chromosome $G(V, E)$, let $G'(V', E')$ be the graph chromosome after such unary operation of graph evolution. It is impossible that

$$E \supseteq E' \text{ and } V \subset V' \quad (1)$$

or

$$E \subseteq E' \text{ and } V \supset V' \quad (2)$$

Proof:

(A) For Eq.(1), we first assume that $V' = V + v_i \equiv V \cup \{v_i\} \supset V$, then

$$E' = \left\{\overline{v_{j_1} v_{j_2}} \middle| v_{j_1}, v_{j_2} \in V\right\} + \left\{\overline{v_j v_i} \middle| v_j \in V\right\},$$

in which $\left\{\overline{v_{j_1} v_{j_2}} \middle| v_{j_1}, v_{j_2} \in V\right\} \subseteq E$ but $E \cap \left\{\overline{v_j v_i} \middle| v_j \in V\right\} = \varnothing$ (empty set). By the connectedness of $G'(V', E')$, $\left\{\overline{v_j v_i} \middle| v_j \in V\right\}$ is nonempty. This contradicts $E \supseteq E'$.

(B) Eq. (2) can be also proven similarly. □

The above lemma explains the impossible conditions in Table 1. The other five conditions for unary operations in Table 1 are then defined as follows.

Reproduction: $E = E' \text{ and } V = V'$.

For mapping to the mutation operation in genetic algorithm, there are two sorts of unary operations in graph evolution are:

Differentiation: defined as the evolved edge set is larger than the original edge set and divided into the following two kinds:

Node differentiation: $E \subset E' \text{ and } V \subset V'$.

Edge differentiation: $E \subset E' \text{ and } V = V'$.

Pruning: is just the reverse conditions of differentiation, where the edge set is shrink: $E \supset E' \text{ and } V \supseteq V'$.

In graph evolution, all these unary operations belong to *mutation operations*, which can be formally defined as:

$$s\left(G(V, E)\right) = G'(V', E'),$$

where $G(V, E)$ and $G'(V', E')$ are both graph chromosomes. Here, the definitions of $s(\cdot)$ consider the *basic mutations*, that is, those operations adding/deleting a single node/edge, as Table 2 listed.

TABLE 2. THE VARIATIONS OF ORDERS/SIZES FOR BASIC MUTATIONS

$s\left(G(V, E)\right) = G'(V', E')$		Order $\lvert V'\rvert$	Size $\lvert E'\rvert$
Pruning	Node Pruning	$\lvert V'\rvert = \lvert V\rvert - 1$	$\lvert E'\rvert = \lvert E\rvert - 1$
	Edge Pruning	$\lvert V'\rvert = \lvert V\rvert$	$\lvert E'\rvert = \lvert E\rvert - 1$
Differentiation	Edge Diff.	$\lvert V'\rvert = \lvert V\rvert$	$\lvert E'\rvert = \lvert E\rvert + 1$
	Node Diff	$\lvert V'\rvert = \lvert V\rvert + 1$	$\lvert E'\rvert = \lvert E\rvert + 1$

IV. BINARY OPERATION

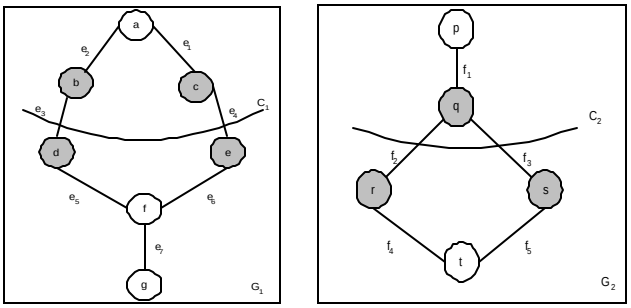
Unlike the mutation operation analyzed above, another genetic operation is so-called *crossover* (or *recombine*), taking two chromosomes as inputs. After a *cut-position* is chosen, each chromosome can be divided into two pieces. Recombining two pieces selected from different piece sets of the *ascendant chromosomes* (or called *parent chromosomes*), two *descendant chromosomes* (or called *children chromosomes*) finally can be created from the ascendant chromosomes by *recombine operation*, as Figure 2 following illustrated.

According to genetic algorithm, some condition has to be imposed on these chromosome pieces to ensure recombine operations, as Definition 2 stated.

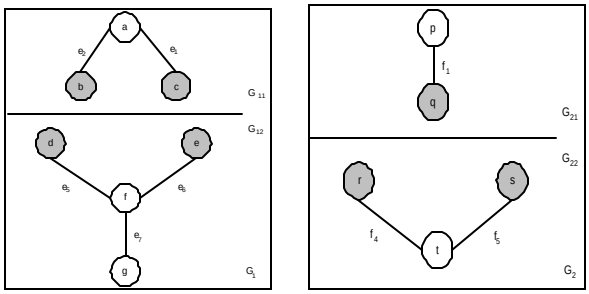
Definition 2. When two or more chromosomes are split in the same cut-position, the chromosome pieces from different parents are called *matchable chromosome pieces*. □

Recombining two matchable chromosome pieces we can form new children chromosomes. When applied to graph evolution, such formulation of binary genetic operation is not so obvious. The first problem is how to determine the cut-position of a graph chromosome. In graph theory, there exist two possible methods for cutting a graph: cut-set and cut-vertex. [8]. By using a *cut-set* C_i, Figure 3 illustrates the matching problem. Hence, the corresponding

match-ability condition of chromosomes is introduced with a slice change in Definition 3.



(a) Graph chromosomes with specific cut-sets



(b) graph chromosome pieces of (a)

Fig. 2. Graph chromosomes and their cutting

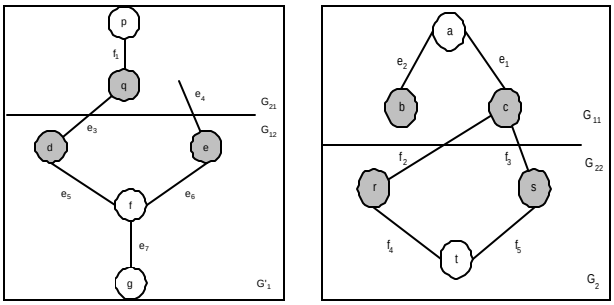


Fig. 3. Matching problems of recombination

Definition 3. Given the cut-position vertices V_{C_i} of graph G_i (i=1,2) split by a cut-set C_i. Two chromosome pieces (V_{C_1}-side and V'_{C_2}-side, or V_{C_2}-side and V'_{C_1}-side) are called *matchable* if $|V_{C_1}| = |V_{C_2}|$ and $|V'_{C_1}| = |V'_{C_2}|$. □

The other cutting method of a graph is using *cut-vertex*. Split through a cut-vertex v_c, a graph G can be divided into two subgraphs. In this case, the corresponding *cut-set* can be treated as a pseudo edge that connects each cut-position vertex pairs, as Figure 4 illustrated.

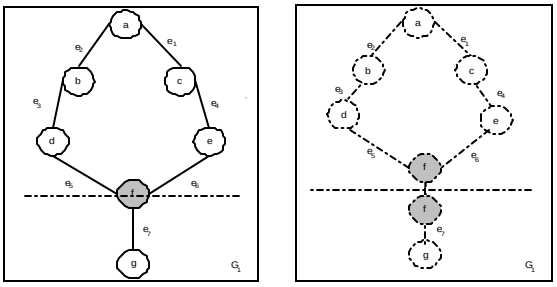


Fig. 4. Cut-vertex and its chromosome pieces

Now, we are going to further investigate the match-ability condition of two complete graph chromosomes. The following lemma describes that graph homeomorphism will ensure such match-ability, and Theorem 1 is also possible to prove that such recombine operator will preserve the homeomorphism of graph chromosomes according to Lemma 2.

Lemma 2. Two graphs are matchable if they are homeomorphic.

Proof: Details will discuss in the complete paper. □

Theorem 1. Given two graph chromosomes with the condition $G_1 \overset{homeo.}{\sim} G_2$. The resultant chromosome comes from the recombine operation:

$$G'(V', E') = p\ (G_1(V_1, E_1),\ G_2(V_2, E_2)),$$

then $G' \overset{homeo.}{\sim} G_1$ and $G' \overset{homeo.}{\sim} G_2$.

Proof: Details will discuss in the complete paper. □

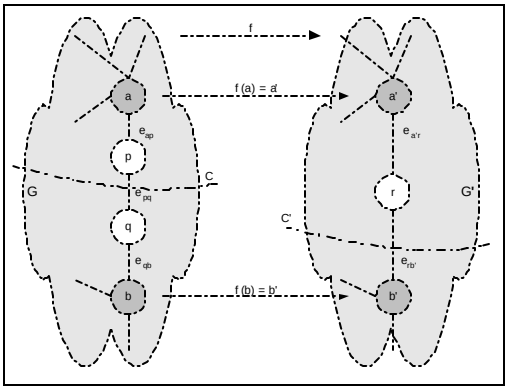


Fig. 5. Cut-sets in homeomorphic chromosome

After the above important properties are analyzed, the

recombine operator p can be defined for a successful operation.

$$G'(V',E') = p\ (G_1(V_1, E_1),\ G_2(V_2, E_2)) \quad \text{or}$$
$$G_1(V_1,E_1)\ p\ \ G_2(V_2,E_2) \rightarrow\ G'(V',E')$$

Different composite operations can be decomposed into several simple operations analyzed previously. According to the rules of simple operations of mutating graph and recombine graphs, an *evolution lattice* might be simply drawn out for helping analyze and even predict the style of a graph chromosome after several generations in future.

V. EXPERIMENT SYSTEM

After the genetic operations in graph evolution have been analyzed and designed in the previous sections, an experimental system will be implemented to demonstrate how to apply graph evolution into the real-world problems. This system can be used not only to test the practicability of graph evolution, but also to evaluate the performance of the example systems.

Experiment and those comparison methods in this paper use the data on educational transitions for a sample of 500 Irish schoolchildren aged 11 in 1967. [21] The data were collected by Greaney and Kelleghan (in 1984) and placed on the Web site http://lib.stat.cmu.edu/datasets/irish.ed. After some erroneous records are eliminated, there are 380 records left as training patterns and 93 as test patterns.

The theory of graph evolution tries to modify the above Kohonen's SOM, including both linear and mesh SOMs. The linear SOM modified by graph evolution is called *Evolutionary SOM* (*ESOM*); whereas, the mesh SOM is modified to *Evolutionary Mesh SOM* (*E-MSOM*). From the viewpoint of classification error, Figure 6 shows out both the classification errors in Kohonen's SOMs (linear SOM/mesh SOM) and their corresponding experiment SOMs modified by graph evolution (ESOM/E-MSOM).

VI. CONCLUSIONS

This paper proposes a new evolution algorithms for graph-form chromosomes, and makes a preliminary research on the graph evolution by using GA and GP as references.

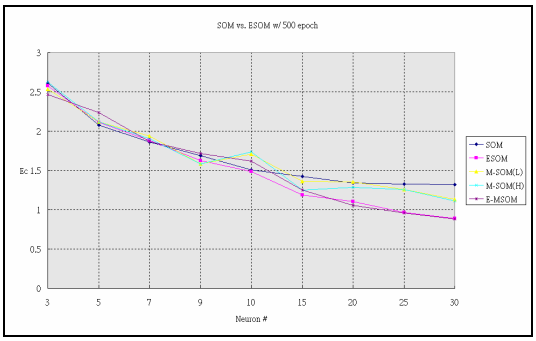


Fig. 6. Comparisons of SOMs and ESOMs

Fundamental analysis of genetic phases of evolution is made. According to those characteristics of graphs in graph theory, genetic operations such as mutation and recombination are investigated. The fitness function for selection in the evolution phase should be also well designed in near future.

Besides the analysis and design of graph evolution, an experiment system is also implemented for evaluating the performance of the systems when the graph evolution is used to replace the original system. Not only the curves of learning rates and outcomes are better in graph evolution, but also the training time needed to solve problem is acceptable.

Furthermore, the relationships between genetic operations and graph complexity might be able to figure out by studying the lattice of graph evolution. Once again, if the lattice of graph evolution can be drawn down, not only the fittest graph chromosomes/patterns (solutions) to environment (problem) might be able to be found out, but even the future trends of the population (composed of graph chromosomes) could also be predictable.

VII. REFERENCES

[1] T. Bäck, Evolutionary Algorithm in Theory and Practice, NY: Oxford University Press, 1996

[2] T. Bäck and H. P. Schwefel, (1993), "An Overview of Evolutionary Algorithms for Parameter Optimization," Evolutionary Computation, Vol. 1, No. 1, MIT Press, 1993, pp.1-23

[3] T. Bäck and H. P. Schwefel, "Evolution Strategies I:

Variants and their computational implementation," Genetic Algorithms in Engineering and Computer Science, pp.111-126, John Wiley & Sons, 1995

[4] T. Bäck and H. P. Schwefel, "Evolutionary computation: an overview," Proceedings of the Third IEEE Conference on Evolutionary Computation, Nagoya, Japan, 1996

[5] Michael Berthold and David J. Hand, (1999), Intelligent Data Analysis, Berlin, Heidelberg: Springer-Verlag, 1999

[6] W. Banzhaf, P. Nordin, R. Keller and F. D. Francone, (1997), Genetic Programming - An Introduction, San Francisco, CA; Dpunkt, Heidelberg: Morgan Kaufmann, 1997

[7] Dingjun Chen, Takafumi Aoki, Naofumi Homma, Tashiki Terasaki and Tatsuo Higuchi, (2002), "Graph-Based Evolutionary Design of Arithmetic Circuits," IEEE Transactions on Evolutionary Computation, Vol. 6, No. 1, Feb. 2002, pp.86-100

[8] G. Chartrand and O. R. Oellermann, Applied and Algorithmic Graph Theory, McGraw-Hill Inc., 1993

[9] Tzi-Dar Chineh, Tser-Tzi Tang and Liang-Gee Chen, "Vector Quantization Using Tree-Structure Self-Organizing Feature Maps," IEEE Journal on Selected Areas in Communication, Vol.12, No.9, December 1994, pp.1594-1599

[10] Maiga Chang, Horng-Jyh Yu, Jia-Sheng Heh (1998), "Evolutionary Self-Organizing Map," Proceedings of the World Conference on Computational Intelligence, (WCCI '98), Anchorage, Alaska, USA, May 4-9, 1998, pp.680-685

[11] Charles Darwin, (1859), On the Origin of Species, London: Murray, 1859

[12] Àgoston Endre Eiben, Robert Hinterding and Zbigniew Michalewicz, "Parameter Control in Evolutionary Algorithms," IEEE Transactions on Evolutionary Computation, Vol.3, No.2, July, 1999, pp.124-141

[13] D. Fogel and A. Ghozeil, "Using fitness distributions to design more efficient evolutionary computations," Proceedings of the Third IEEE Conference on Evolutionary Computation, Nagoya, Japan, 1996, pp. 11-19

[14] M. Gen and R. Cheng, Genetic Algorithms and Engineering Design, John Wiley & Sons, 1997

[15] Al Globus, J. Lawton and T. Wipke, (2000), "JavaGenes: Evolving Graphs with Crossover," http://www.nas.nasa.gov/~globus/papers/JavaGenes/paper.html, June 2, 2002 (not publish yet)

[16] Al Globus, John Lawton and Todd Wipke, (2001), "Graph Crossover," Proceedings of the Genetic and Evolutionary Computation Conference, GECCO-2001, Edited by L. Spector, E. Goodman, A. Wu, W.B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. Garzon and E. Burke, San Francisco, CA:

[17] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Reading, MA: Addison Wesley, 1989

[18] K. Hitomi, Manufacturing Systems Engineering, 2nd ed., London: Taylor & Francis, 1996

[19] J. Holland, Adaptation in Neural and Artificial Systems, Ann Arbor : University of Michigan Press, 1975

[20] J. H. Holland, (1992), Adaptation in Natural and Artificial Systems, Cambridge, MA:MIT Press, (Reprint of Univ. of Michigan Press), 1992

[21] Hoeting Jennifer, Raftery Adrian E. and Madigan David, "Method for simultaneous variable selection and outlier identification in linear regression," Computational Statistics & Data Analysis, Vol. 22, No. 3, July 15, 1996, pp.251-270

[22] Teuvo Kohonen, "Self-Organization and Associative Memory," Lecture Notes in Information Science, Vol. 8, NY: Springer-Verlag, 1984

[23] John R. Koza, (1989), "Hierarchical Genetic Algorithms Operating on Populations of Computer Programs," in Proceedings of the 11th International Joint Conference on Artificial Intelligence, Edited by N. S. Sridharan, San Mateo, CA: Morgan Kaufmann, 1989, pp.768-774

[24] Z. Michalewicz, "Evolutionary computation: practical issues," Proceedings of the Third IEEE Conference on Evolutionary Computation, Nagoya, Japan, 1996, pp. 30-39

[25] Ari S. Nissinen and Heikki Hyötyniemi, (1998), "Evolutionary Self-Organizing Map," 6th European Congress on Intelligent Techniques and Soft Computing, (EUFIT '98), Aachen, Germany, Sep. 7-10, 1998, pp.1596-1600

[26] S. Raman and L. Patnaik, (1997). "Optimization via Evolutionary Processes," Advances in Computers, Edited by M. V. Zelkowitz, Vol. 45, San Diego: Academic Press, 1997, pp. 156-196

[27] H. P. Schwefel, Evolution and Optimum Seeking, NY: John Wiley & Sons, 1994

[28] N. Singh, System Approach to Computer-Integrated Design and Manufacturing, NY: John Wiley & Sons, 1996