International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075, Volume-8 Issue-8S3, June 2019

Efficient Encoding Scheme in Genetic Algorithm for Requirement Optimization

Rajesh Kumar, Rakesh Kumar

Abstract*: Requirements analysis is the initial step of the software development process. Requirement selection is an engineering process to select a best set of system requirements for implementation. In this paper, Genetic algorithm is used as an optimization technique to optimize the requirements. Every search and optimization algorithm needs a technique to represent the probable solutions to a particular problem. In order to make the use of genetic algorithm, one should be able to represent the solutions in the form of chromosomes so that crossover, mutation etc. operators can be applied to generate the new solutions. In this paper, limitations of a number of Encoding scheme has been identified and a Encoding scheme is proposed for chromosome for identifying optimum requirements.

Index Terms: Chromosome, Encoding, Genetic Algorithm Requirement, Optimization.

I. INTRODUCTION

Requirements analysis is the first phase of the software development life cycle and it's one of the main concern of software engineering. System requirement selection is the engineering process to find a minimum set of system requirement for implementation. Additionally, in real life problems, the requirements selection suffers from complication due to varying interest of heterogeneous users. To find the optimal set of requirements, there is a strong need of optimization techniques. Meta-heuristic algorithms, such as tabu search, simulated annealing and genetic algorithm have been applied to a wide range of optimization or search problems. In this paper, the use of genetic algorithm has been discussed in requirement engineering .

II. GENETIC ALGORITHM

Genetic algorithm (GA) is categorized under evolutionary algorithms. They are primarily used for solving optimization problems. In big problem, state space representation is characterized by a set of objects; each of them has distinct parameters. The objective of optimization problem, working on above mentioned parameters and optimize them. All optimization technique needs a representation method which depictions the solutions to a specific problem. Likewise, GA work on coding space and solution space alternatively. Genetic algorithms are inspired from biology, so having two spaces is supported by natural evidence. In nature, coding space refers to the genotypic space and phenotypic space is referred by solution space. This had been clearly investigated by Mendel in 1866[1]. A transformation exists between genotype and phenotype, also called mapping, which uses the genotypic information to design the phenotypic trait. A chromosome specify to a string of positive length where all the genetic data of an individual is stored. All chromosomes consist of many allelomorph. Alleles are the littlest data items in a chromosome [2]. If a phenotypic property of an individual, like its eye color is determined by one or more allelomorphs, then these allelomorphs together are denoted as gene as shown in Figure 1.

$$\boxed{010101\boxed{10}101}$$

Figure 1. Representation of Gene-Allele in chromosome.

A gene is a location on a chromosome, which is responsible for a particular character state property [3].
Problem representation in GA needs some Encoding scheme. Desirable trait of Encoding are (i) it should be able to represent all possible phenotypes, (ii) it should encode no infeasible solutions, (iii) it should be unbiased, (iv) decoding from phenotypic trait to genotype should be easy, and (v) problem should be represented at the correct level of abstraction.

III. TYPES OF ENCODING

A. *Binary Encoding*

It is most familiar representation of chromosome in GA. A Binary string is described by using a binary alphabet {o, 1}. Each functional variable is encoded in a binary alphabet of a certain length l_i, described by the user [4,5,6]. Thereafter, a complete 1-bit string is formed by connecting all substrings together. Thus, the length L of string in GA has:

$$S = \sum_{i=1}^{N} Si \qquad (1)$$

such that N represent the number of object variables.

Revised Manuscript Received on Date
Rajesh Kumar, Dept. of Computer Science & Applications, K.U., Kurukshetra, Haryana, INDIA.
Prof. Rakesh Kumar, Dept. of Computer Science & Applications, , K.U., Kurukshetra, Haryana, INDIA.

Binary Encoding allows for a higher degree of parallelism. It contains more schemas than decimal coding. In spite of these benefits, they are abnormal and clumsy for many problems. They are prone to arbitrary orderings [7,8]. Binary Encoding does not supply the collection of options required to deal with the variety of problems endure in business, science and engineering. In this encoding, distinct bits have disparate significance and hamming distance between two successive integers is generally not equivalent to one [9,10].

B. Gray Encoding

Normal binary Encoding representation of the object may gradual convergence of a GA. Expending the number of bits in the binary representation amplify the problem [10,11]. Gray Code can prevent this problem by redefining the binary strings so that consecutive numbers have a Hamming distance of one [12]. Gray codes accelerate convergence time by keeping the algorithms attention on converging toward a solution [13].

C. Floating Point Encoding

Floating-point Encoding (FPE) scheme was designed by Deb for continuous variables [14]. In FPE scheme, distinct genes named as M represent mantissa and E represent exponent of a Floating-point parameter. For a multi--parameter optimization problem, a common gene has three elements, as opposed to two in earlier Encoding schemes. The three elements are (i) parameter ID number, (ii) mantissa or exponent declaration, and (iii) its value. Various experiments have shown that the floating point Encoding is simple to implement, as it does not require any conversion mechanism that converts bit string to real value. Various experiments have shown that the genetic algorithm with floating point Encoding is simpler for implementation and faster than genetic algorithm with binary representation. This is because in case of binary implementation, the algorithm must have conversion mechanism that converts bit string to real value. Such mechanism is not required in case of floating point Encoding. It has been found that algorithm with floating point Encoding gives better results than algorithm with binary Encoding as it can easily incorporate various constraints.

D. Permutation Encoding

Permutation Encoding (PE) is used in Ordering problems, such as traveling salesman problem etc. Given N rare objects, N! permutations of the object exist. In Permutation scheme, every chromosome is a string of numbers that represent a position (number) in a sequence. In certain cases random generated numbers between o and 1 are also used to code the problem. Then to decode the solution these generated values are used as sort keys. Sample representation for two chromosomes is

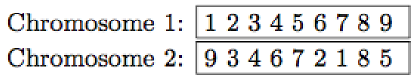
Chromosome 1: 1 2 3 4 5 6 7 8 9
Chromosome 2: 9 3 4 6 7 2 1 8 5

Figure 2. Representation of chromosome in PE

Permutation issues can not be handled using the same generic recombination and mutation operators that are enforced to parameter optimization problems. Permutations are also substantial for scheduling applications, variants of which are also often Non Deterministic time Polynomial (NP) complete. This Encoding is likewise known as path representation [15].

E. Value Encoding

Value coding scheme can be used in problems where some more arduous values such as real numbers are used [16]. Use of simple Binary Encoding for such type of problems would be very difficult. In this Encoding scheme, all chromosomes is a array of some values that can be anything connected to the problem, such as real numbers, characters or any objects. It is often necessary to develop some specific crossover and mutation techniques for these chromosomes [17]. Sample representation is:

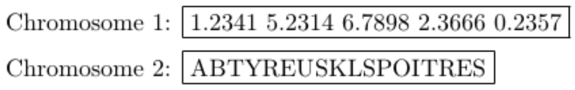
Chromosome 1: 1.2341 5.2314 6.7898 2.3666 0.2357
Chromosome 2: ABTYREUSKLSPOITRES

Figure 3. Representation of chromosome in Valve encoding

F. Tree Encoding

It is used generally for genetic programming. In this tree Encoding scheme all chromosome is a Tree of any objects, such as behavior or commands in programming language [18]. The representation space is described by characterizing the set of behavior and terminals to label the verities in the tree. Trees support rich representation that is adequate to represent computer-programs, analytic functions, and dynamic length structure, even computer-hardware. Parse tree is a well-known representation for emerging executable structures [4]. Parse tree include natural recursive definition, which allows for variable sized structures. In this representation, the elements of the parse

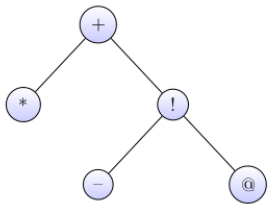

Figure 4. Tree Encoding Representation

tree determine the power and fitness of the representation. As a result of cyclic nature of parse trees, iterative computations are not easily characterized.

To classify the stopping criteria for such problems is very difficult. So, the derived function is evaluated within a hidden loop that re-executes the evolved function until some predefined stopping criteria is reached.

G. Grammar based Encoding

Grammar based Encoding is used in problems dealing with optimization of networks [19]. The global motions of automata networks (such as artificial neural networks) are a function of its topology and the choice of automata used. Evolutionary computation can be enforced to the optimization of the parameters, but their computing cost is restrictive unless they operate on a solid representation. Earlier direct coding was used as it was simple. This Encoding schemes use construction for genes, the verbalization of which forms the phenotype [6]. The circumlocution of this Encoding admit for a many-to-one mapping with a corresponding richness in possible genotypic representations. Graph grammars provide such a representation by allowing network regularities to be efficiently captured and reused.

IV. NEED OF NEW ENCODING SCHEMES

The effectiveness of software system, it totally depends on the measurement in what way both the demands of its users and its operational environment were being met and those needs were included in the system requirements. In this view, the requirements engineering (RE) could be described as a process by which the requirements are driven thus, successful RE included the following aspects:

- Understanding the various needs of users, customers and other stakeholders
- Understanding the approach in which the system would be developed
- Making sure that the documented requirements were consistent.

In the context of a big software, the early stage of the requirement engineering is very demanding, as its resulted decisions are both crucial and difficult; and that is due to the different types of users and their varying interest of system interaction.

Moreover, these decisions have a abiding impact on the software system.

Thus, selection of requirements from heterogeneous requirements is a decision-making process that empowered system managers to focus on the deliverable software that added most value to a system's outcome. To find the optimal set of requirements, justify that their is a need of optimization. As discussed earlier, in this paper genetic algorithm is used to optimize the requirement selection process. In order to use the genetic algorithm, an Encoding scheme is used to represent the solution in the form of chromosome.

A. Design principle of Encoding Scheme

In order to design a new Encoding scheme, designer must understand the basic principal of building blocks(BB). According to Goldberg [2o], design of encoding scheme has a strong impact on the performance of Genetic algorithm and should be chosen carefully. Goldberg purpose two basic design principal namely:

1. The principle of meaningful building blocks
2. The principle of minimal alphabets

The first principle states that user should preferred an encoding coding scheme such that the building blocks of a primary problem are limited and analogously unrelated to building blocks at other locations. The principle of significant building blocks is directly inspired by the \d schema theorem [2]. If schemata are deeply fit, precise and of low order, then their numbers rapidly increase over the generations.

The other principle states that the user should select the minimal element that authorizes an expression of the problem so that the number of exploitable schemas is maximized [21]. The principle of minimal elements tells us to maximize the promising number of schema by compressing the cardinality of the elements. When using minimal elements the number of possible schemata is maximal. This is the reason why Goldberg advises to use bit string representations, because deep quality schemata are more hard to find when using elements of bigger cardinality.

V. ANALYSIS OF EXISTING ENCODING SCHEMES

Depending on the design of Encoding scheme, it can be divide into two class, known as, 1-dimensional and 2-dimensional. Binary, Value, Real value and Permutation Encoding are 1-dimensional and Tree Encoding is 2-dimensional Encoding techniques [3].

Analyzing these Encoding schemes, everyone can conclude that characters represented by permutation Encoding are position reliant. In the Binary Encoding scheme and real value Encoding scheme, the characters are value dependent.

The two different aspect classified by studying unlike Encoding schemes are value and locus in the chromosome. Thus, factors like value and locus should be kept in mind while Encoding a solution for a certain problem. Binary Encoding is the familiar arrangement and backings various types of crossover activities.

Integer and FPR have defined application rely upon the problem. Permutation encoding scheme is used to show certain order in chromosomes. It allows inversion operator but does not support simple crossover operator like one point crossover, N point crossover. Special crossover operators like Partially-mapped crossover (PMX), Order crossover (OX) or Cycle crossover (CX) is used for this representation making its use absolutely complex.

VI. GOLDBERG CATEGORIZATION OF ENCODING

Goldberg has concluded in his work that fitness function for a particular Encoding technique is dependent on two factors-- **value** and **order** [22]. Three different categories of Encoding can be grouped depending on fitness evaluation factors such as:

- ✓ Encoding schemes where fitness depends on only value: f(**V**). Eg: Value Encoding
- ✓ Encoding schemes where fitness depends on order and value: f(**V,o**). Eg: Binary Encoding
- ✓ Encoding schemes where fitness depends on only order: f(**o**). Eg: Permutation Encoding.

VII. RESULTS AND PROPOSED ENCODING SCHEME

So, the extant Encoding schemes belong these three classes and accordingly fitness function can be devised. In the existing Encoding schemes discussed in the previous sections, the length of the chromosome is fixed, and such Encoding schemes are not suitable for the representation of the requirements because different users view the problem from different dimensions. Consequently, if the requirements of a user are represented using chromosome wherein genes indicate the requirements on a thread then the size of chromosome will not be fixed it will varying from one user to another. Their requirements are varying, some are common and some are altogether different. It is a big challenge for software engineers to represent the requirements (gathered from the different users), in the form of a chromosome because the heterogeneity of the requirements. Consequently, there arises a need to bring to light a new Encoding scheme that is independent of above said two factors i.e **value** and **order**. In this new Encoding scheme, Requirements are stored in the form of a **SET** to maintain the uniqueness and a number is assigned to every unique requirement.

A. *Prerequisites for new Encoding scheme*

It is assumed that there are **N** users and **M** possible system requirements are gathered from users, out of these **M** requirements some are common and some are altogether different. Every requirement is assign a unique number on the basic of their uniqueness.

It is assumed that there is a **SET** of users, **N**={N_1,$N_{2,}$……….$N_{n\}}$ and a **SET** of possible system requirements, **R** = {{R_1,R_2 ,……………………R_m}

In this new Encoding scheme all the requirements gathered from a user are represented as a chromosomes in raw form. In figure 3, chromosome 1 represents the requirements gathered from user 1 and so on.

$$\text{Chromosome 1: } \boxed{R_1, R_3, R_6, R_8, R_9, R_{12}, R_{15}}$$

$$\text{Chromosome 2: } \boxed{R_1, R_3, R_4, R_6, R_7, R_{12}}$$

$$\text{Chromosome 3: } \boxed{R_1, R_2, R_5, R_8, R_{10}, R_{16}, R_{20}, R_{23}}$$

Figure 5. Representation of the requirements in the form of a chromosome.

VIII.

IX. CHALLENGES FOR NEW ENCODING SCHEME

A. *Fitness function for new Encoding scheme*

The next challenge is to find the fitness function for proposed Encoding scheme because no mathematical function is available to calculate the fitness valve of a chromosome. As a consequence, human based oracle [23] can be used as a fitness function. Therefore, human based computation is an alternate fitness function.

B. *Crossover Operators for Different Representations*

Binary Encoding is the simplest method of representing chromosome. In crossover operation, it takes two chromosomes as parent and generates two offspring chromosomes. Three distinct forms of crossover suitable for binary Encoding are one point crossover, N point crossover and Uniform crossover. The size of chromosome in new Encoding scheme is not fixed. Consequently, different crossover operators used in fix length binary Encoding scheme can not be used in this new Encoding scheme. Chromosomes having Real value or FPR endure Arithmet*ic* crossover. This crossover builds a new allele at each gene location in the off-spring. The value of new allele goes between the values of the ancestor alleles.

The crossover operator used in valve Encoding can not be used in new Encoding scheme. In the value Encoding, every gene represents some value but in the proposed Encoding, every gene represents a requirement. Encoding scheme acquiring integer values in their representation also execute the same set of crossover operations as executed by Binary Encoding such as One point crossover, N point crossover, Uniform crossover.

In permutation Encoding, the fitness value of a chromosome depends on the order of genes but new Encoding scheme is independent of this factor. In view of the above facts, it is concluded that selection of crossover operator is a big
challenge for this proposed Encoding.

C. *Mutation Operators for Different Representations*

Inversion of bit is very elementary mutation used for binary Encoding. In this, bit 1 inverted by 0 and vice-versa. The number of bits that invert rely upon the mutation rate. For Real value or FPR, there exists uniform mutati0n, Gaussian mutation and Boundary mutati0n. For permutation representation, there are collection of mutation operators like Swap, Scramble and Inversion [23]. Selection of mutation operation is another challenge for new novel Encoding scheme.

X. CONCLUSION AND FUTURE WORK

The intention of the current research paper was to optimize the requirement engineering process using meta-heuristic. In order to use meta-heuristic such as GA, first step is to represent all the possible solutions of a problem in the form of a chromosome. In this current paper, it has been identified that the existing Encoding schemes are not suitable to optimize the requirements. A new Encoding scheme for requirement 0ptimization has been proposed which can deal with the chromosomes of varying lengths. It is hoped that the set of challenges discussed in the previous section would serve to stimulate further research in this area.

REFERENCES

1. Gregor Mendel. Versuche über pflanzenhybriden. Verhandlungen des naturforschenden Vereines in Brunn4: 3, 44, 1866.
2. John H. Holland. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI, 1975. second edition, 1992.
3. Franz Rothlauf and David E. Goldberg. Representations for Genetic and Evolutionary Algorithms. Physica-Verlag, 2oo2.
4. Thomas Back, David B. Fogel, and Zbigniew Michalewicz, editors. Handbook of Evolutionary Computation. IOP Publishing Ltd., Bristol, UK, UK, 1st edition, 1997
5. Thomas Bäck, David B Fogel, and Zbigniew Michalewicz. Evolutionary computation 1: Basic algorithms and operators. CRC press, 2o18.
6. E Vonk, LC Jain, and R Johnson. Using genetic algorithms with grammar encoding to generate neural networks. In Proc. 1995 IEEE Int. Conf. Neural Networks, Part 4 (of 6), pages 1928–1931, 1995.
7. Melanie Mitchell. An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA, USA, 1996.
8. Yong-Yi FanJiang and Yang Syu. Semantic-based automatic service composition with functional and non-functional requirements in design time: A genetic algorithm approach. Information and Software Technology, 56(3):352–373, March 2o14.
9. Agoston E Eiben, P-E Raue, and Zs Ruttkay. Genetic algorithms with multi-parent recombination. In International Conference on Parallel Problem Solving from Nature, pages 78–87. Springer, 1994.
10. Guang-Zheng Zhang and De-Shuang Huang. Prediction of inter-residue contacts map based on genetic algorithm optimized radial basis function neural network and binary input encoding scheme. Journal of computer-aided molecular design, 18(12):797–810, 2004.
11. Randy L. Haupt and Sue Ellen Haupt. Practical Genetic Algorithms. John Wiley & Sons, Inc., New York, NY, USA, 1998.
12. Herbert Taub and Donald L Schilling. Principles of communication systems. McGraw-Hill Higher Education, 1986.
13. Richard A Caruana and J David Schaffer. Representation and hidden bias: Gray vs. binary coding for genetic algorithms. In Machine Learning Proceedings 1988, pages 153–161. Elsevier, 1988.
14. Mohamed Wiem Mkaouer, Marouane Kessentini, Mel Cinnide, Shinpei Hayashi, and Kalyanmoy Deb. A robust multi-objective approach to balance severity and importance of refactoring opportunities. Empirical Software Engineering, 22(2):894–927, April 2017.Timothy Starkweather, S McDaniel, Keith E Mathias, L Darrell Whitley, and C Whitley. A comparison of genetic sequencing operators. In ICGA, pages 69–76, 1991.
15. Chahine Koleejan, Bing Xue, and Mengjie Zhang. Code coverage optimisation in genetic algorithms and particle swarm optimisation for automatic software test data generation. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC ’15), pages 1204–1211, Sendai, Japan, 25-28 May 2015. IEEE.
16. Meng Qingchun. An approach on genetic algorithm with symmetric codes. Acta Electronica Sinica (China), 24(10):27–31, 1996.
17. JF Aguilar Madeira, HL Pina, and HC Rodrigues. Gatopology optimization using random keys for tree encoding of structures. Structural and Multidisciplinary Optimization, 40(1-6):227, 2010.
18. Earl T. Barr, Mark Harman, Yue Jia, Alexandru Marginean, and Justyna Petke. Automated software transplantation. In Proceedings of the 2015 International Symposium on Software Testing and Analysis (ISSTA ’15), pages 257–269, Baltimore, USA, 14-17 July 2015. ACM.
19. David E Goldberg. Genetic algorithms. Pearson Education, India, 2006.
20. David E Goldberg. Messy genetic algorithms: Motivation analysis, and first results. Complex systems, 4:415–444, 1989.
21. David E Goldberg and John H Holland. Genetic algorithms and machine learning. Machine learning, 3(2):95–99, 1988.
22. AE Eiben and JE Smith. Introduction to evolutionary computing (natural computing series). 2008.
23. Yuen, M.-C., Chen, L.-J., and King, I. (2009). A survey of human computation systems. 2009 International Conference on Computational Science and Engineering, 4:723–728

AUTHORS PROFILE

Rajesh Kumar obtained his B.Sc. Degree, Master's Degree (Master of Computer Applications) from Kurukshetra University, Kurukshetra. Currently, He is an Assistant Professor in the Department of Computer Science and Application's, in Chhaju Ram Memorial Jat College, Hisar, Haryana, INDIA. His research interests are in Genetic Algorithm, Theory of Automata, Software Engineering, Artificial Intelligence, Design and Analysis of Algorithm and Linux.

Rakesh Kumar obtained his B.Sc. Degree, Master's Degree -- Gold Medalist (Master of Computer Applications) and PhD (Computer Science \& Applications) from Kurukshetra University, Kurukshetra. Currently, He is Professor & Chairperson in the Department of Computer Science and Applications, Kurukshetra, University, Kurukshetra, Haryana, INDIA. His research interests are in Genetic Algorithm, Software Testing, Artificial Intelligence, and Networking. He is a senior member of International Association of Computer Science and Information Technology (IACSIT).