COMPARATIVE STUDY OF PHONETIC PATTERNS IN CHHANDA SHASTRA AND BINARY PATTERNS IN GENETIC ALGORITRHMS

Sugandha Dani*

Abstract: *Chhanda Shastra of Pingalacharya is a well-known classical Sanskrit treatise on prosody. It thus deals with the rhythmic syllable arrangements in poetic meters. According to Chhanda-Shastra, every pronounced letter is either Laghu (ᑊ / ⁻/ 0 / ल) or Guru (` / ऋ / 1 /ग).*

Following table depicts an example of three letter-words with the eight possible permutations of its syllables based on above Sutra. Using the mathematical axiom the resemblance of corresponding symbolic representation with binary sequence is also अंकानां वामतैं गतिः indicated.

Laghu / Guru Sequence	Example (Devnagri)	Symbolic Representation	Equivalent Binary Sequences
ल ल ल	नयन	न । । ।	0 0 0
ल ल ग	सरला	स । । ऽ	0 0 1
ल ग ल	जपान	ज । ऽ ।	0 1 0
ल ग ग	यमाणां	य । ऽ ऽ	0 1 1
ग ल ल	भारत	भ ऽ । ।	1 0 0
ग ल ग	राजते	र ऽ । ऽ	1 0 1
ग ग ल	तापाय	त ऽ ऽ ।	1 1 0
ग ग ग	मार्गुर्यम्	म ऽ ऽ ऽ	1 1 1

One of the great visionary Pingalacharya thus presents first known description of patterns of binary sequences in the context of syllable arrangements through poetic meters. Later the 10th century mathematician Halayudha wrote a commentary on the ***Chhandas Shastra*** *and expanded it. Halayudha's commentary includes a presentation of meruprastāra or the Pascal's triangle as well as the basic ideas of Fibonacci number (called mātrāmeru in Pingala's text).*

The main topic of Chhandas consists of two types: Vritta (based on number of letters) and Jati (based on matra or phonetics). Thus a single Jati may have multiple Vritta. For example, an Anushtubha is a Jati. Its different possible syllable arrangements are comparable with evolutionary binary sequences appearing in Genetic Algorithm. Such arrangements could be

utilized to generate newer binary sequences, which resemble the solution list generated for multi parameter optimization problem or population of new solutions in genetic algorithm.

A genetic algorithm (GA)is a search technique used in computing to find exact or approximate solutions to optimization and search problems In other words, genetic algorithms are computer algorithms that search for good solutions to a problem from among a large number of possible solutions. The present paper discusses innovative developments through such meaningful comparison.

Keywords: *Genetic Algorithm, fitness function, selection, crossover, mutation, greedy algorithm, minimal spanning tree.*

---

*Assistant Professor, Department of MCA, Priyadarshini College of Engineering, Nagpur

INTRODUCTION

GENETIC CODING

The genetic code [9] is the set of rules by which information encoded in genetic material (DNA or RNA sequences) is translated into proteins (amino acid sequences) by living cells. The code defines a mapping between tri-nucleotide sequences, called codons, and amino acids. A triplet codon in a nucleic acid sequence usually specifies a single amino acid (though in some cases the same codon triplet in different locations can code unambiguously for two different amino acids, the correct choice at each location being determined by context). Because the vast majority of genes are encoded with exactly the same code, this particular code is often referred to as the canonical or standard genetic code, or simply the genetic code, though in fact there are many variant codes.

BASIC IDEA BEHIND GENETIC ALGORITHMS [10]:

A genetic algorithm (GA)[6] [7]is a search technique used in computing to find exact or approximate solutions to optimization and search problems In other words, genetic algorithms in NP hard problem[8] are computer algorithms that search for good solutions to a problem from among a large number of possible solutions. They were proposed and developed in the 1960s by John Holland [9] at the University of Michigan. These computational paradigms were inspired by the mechanics of natural evolution, including survival of the fittest, reproduction, and mutation. Some of the applications of GAs include optimization, automatic programming, machine learning, economics, immune systems, population genetic, and social system.

Genetic algorithms are implemented via computer simulation. A population of abstract representations (called chromosomes or the genotype of the genome) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary strings, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations have been produced, or a satisfactory fitness level has been reached

for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached.

BASIC ELEMENTS OF GAS[11]

Most GAs methods are based on the following elements, populations of chromosomes, selection according to fitness, crossover to produce new offspring, and random mutation of new offspring.

Chromosomes

The chromosomes in GAs represent the space of candidate solutions. Possible chromosomes encodings are binary, permutation, value, and tree encodings.

Fitness function[12]

GAs requires a fitness function which allocates a score to each chromosome in the current population. Thus, it can calculate how well the solutions are coded and how well they solve the problem.

Selection[13]

The selection process is based on fitness. Chromosomes that are evaluated with higher values (fitter) will most likely be selected to reproduce, whereas, those with low values will be discarded. The fittest chromosomes may be selected several times, however, the number of chromosomes selected to reproduce is equal to the population size, therefore, keeping the size constant for every generation.

IMPORTANT STEPS OF GAS INCLUDE:

1. Start: Randomly generate a population of N chromosomes.
2. Fitness: Calculate the fitness of all chromosomes.
3. Create a new population:

a. Selection: According to the selection method select 2 chromosomes from the population.

b. Crossover: Perform crossover on the 2 chromosomes selected.

c. Mutation: Perform mutation on the chromosomes obtained.

4. Replace: Replace the current population with the new population.
5. Test: Test whether the end condition is satisfied. If so, stop. If not, return the best solution in current population and go to Step 2.

Each iteration of this process is called Generation.

COMPARISON AND APPLICATION ANUSTUBH WITH GENETIC ALGORITHM:

As we all know, genetic algorithms may or may not result in accurate answer. This is because of maximum no of population is reached and still satisfactory answer is not derived. Secondly, if no of a population is increased then processing may require more computation time.

Proposed technique suggests use of phonetics patterns observed in Vritta and Chhanda. While pronouncing particular shloka or poem written using specifc vritta or chhhand, they follow certain rules of writing and pronouncing syllables, which makes possible combinations reduced by no of fixed syllables. For example in anustubh chhand, 5th and 6th positions are fixed as laghu and guru respectively. On the same lines we propose to fix possible optimum sub solutions and thereby obtaining accurate solutions with large no of populations without compromising on speed of execution.

Example:

Classic example of Travelling Salesman is taken for finding optimum solution using GA. If we assume that a salesman has to travel 8 different cities such that he has to travel minimum distance. In this case if we can find minimum distance while processing this problem and every time fixing a place to store these solutions then while further processing, mutation or crossover can take place only between remaining chromosomes, which in turn can reduce the processing time. In case if further optimum solution is reached then it can be replaced with previous fixed values. Doing this continuously, we can reach optimum solution within reasonable time.

COMPARATIVE BINARY PATTERNS

The classical Sanskrit text of Pingalacharya Chhandas-Shastra and its mathematical interpretation given by Halayudha report evolution of various variants of patterns of binary sequences which bear striking resemblance with schema and fitness function of genetic algorithm. Hence there is a scope to carry out innovative developments through critical comparative study and research. This paper illustrates such scope through the classic example of Travelling Sales-person who has to visit N number of cities in such a way that the total distance travelled is minimum out of M possible solutions. Using combination of genetic algorithm with patterns of vritta / chhand (where some of the bits are fixed) optimum solution may be approached. We have assumed anustubh for supporting this

example. Eight random solutions are selected as a first set of chromosomes and as per anustubh's rule the minimum distance and maximum distance from selected 8 solutions are stored in two temporary variables. Then following the genetic algorithm-iterations continue selecting further solutions and making crossover and mutations to produce next generation of solutions. These values are compared with minimum distance available after every iteration. Replacement takes place when still smaller solution is noticed. The search for new generation continues. The crossover / mutation takes place with smaller element so that optimum (In this case minimum) solution can be generated without compromising on processing time.

TRAVELLING SALESMAN PROBLEM (TSP)

The Traveling Salesman Problem (TSP) is one of the important subjects which have been widely addressed extensively by mathematicians and computer scientists.

It is a permutation problem with the objective of finding the path of the shortest length (or the minimum cost) on an undirected graph that represents cities or node to be visited. The traveling salesman starts at one node, visits all other nodes successively only one time each, and finally returns to the starting node. i.e., given n cities, named {c1, c2, ..., cn}, and permutations, σ1, ..., σn!, the objective is to choose σi such that the sum of all Euclidean distances between each node and its successor is minimized.

EXISTING METHODS TO SOLVE TSP

- Greedy Algorithm – sort distances between cities in ascending order and selects in same sequence.
- Nearest Neighbour Method – Select city arbitrarily, then select nearest city.
- Minimal Spanning Tree – Select edges (route connecting two cities) with smallest distance, then check distance to neighbouring city, select smallest and add to sum.

DISADVANTAGES OF TRADITIONAL ALGORITHMS

- Greedy algorithm – Can give feasible solution but not necessarily optimum.
- Nearest Neighbour – Choosing small distances for first few cities can leave only large distances to be added at last.
- Minimal Spanning Tree – Backtracking used to select untraveled edges can lead to larger computational time.

WHY TSP USING GA

- To find optimum solution for TSP is the application of exhaustive enumeration and evaluation.
- The procedure consists of generating all possible tours and evaluating their corresponding tour length.
- If we could identify and evaluate one tour per nanosecond (or one billion tours per second), it would require almost ten million years (number of possible tours = 3.2×1023) to evaluate all of the tours in a 25-city TSP."

USE OF CHHANDA PATTERN IN TSPGA

- Arrangement of binary sequences in anustup
- We have considered only two constraints i.e. fifth place fixed - laghu and sixth – guru
- Randomly chosen chromosomes for crossover with minimum distance between two cities placed at 5th position and maximum distance between cities at 6th position.

Example: Traditonal GA

- Cities to be travelled by salesman
- Parent 1: AB BC CD DE EF FG GH HI
- Parent 2: AD DF FB BE EC CH HG GI

  Mask : 10 01 10 00 00 01 10 01
- Child 1 : A_ _C C_ __ __ _G G_ _I

  (mask for parent 1)
- S1 : AD DC CB BE EF FG GH HI

GA using Anustup

- Input cities and distances between them.
- Arrange distances in ascending order.
- Place smallest distance at 5th position and largest at 6th position.
- Create mask for 5th and 6th position, so that they should not get replaced in crossover.
- Starting from 5th position go back till possible combination of city.
- Starting from 6th position move forward till possible combination of city.
- Place remaining pairs at possible places.
- Apply crossover function.

- Store the sequence as feasible solution in search space.
- Continue producing new generations using various permutations of cities.
- In every iteration Rearrange cities with new smallest and new largest values.
- Mark cities with smallest distance as traversed.
- Compare if distance between two cities is smaller than distance placed at 5th position.
    - If yes, then replace it at 5th position.
- Check if all possible combinations are made.
    - If yes, then compare distance with 6th position
        - If distance is smaller than 6th position
        - Replace it with 6th position
- Repeat steps until no smaller value is available.
- Repeat through step 1 until all smallest distances from adjacency matrix are covered.

Example: GA using anustup constraint

- While using anustup constraint, first the distances are arranged in such a way that 5th place holds smallest distance and 6th position holds largest. And then they are masked for crossover.
- Parent 1: AB BC CD DE EF FG GH HI
- Parent 2: IF FD DB BC CH AG GI HE

  Mask : 00 00 00 01 11 11 10 00
- Child 1 : __ __ __ _C CH AG G_ __

  (all except C, H, A, G missing)
- S1 : IB BD DE EC CH AG GH HF

APPLYING ANUSTUP FOR FINDING OPTIMUM SOLUTION

- Store the sequence of cities in search space (solution set).
- Continue finding solutions as per given size of populations.
- Apply anustup again to find smallest solution by randomly selecting 8 solutions and replacing smaller and larger solution.
- Repeat until best solution is found or no possible combination is left.

CONCLUSION:

Further research work is aimed at correlating such binary patterns and their variants appearing in different Chhandas and in genetic algorithm. A critical study would be carried out on the patterns of repetitions of syllables in Vritta and Chhandas and how the same can be inculcated in finding a suitable fitness function for genetic algorithm and how it will be beneficial in optimizing the search Finally, an analysis of the various algorithms will be done with a view to introduce innovative developments.

- Every generation is thus obtained using further minimum values.
- New generation will not be generated with larger values than previous generations.
- Chances of selecting repetitive parents for crossover are reduced, as every traversed city is marked.
- Chances of getting optimum best solution are increased.

REFERENCES:

[1] Amulya Kumar Bag, 'Binomial theorem in ancient India', *Indian J. Hist. Sci.* 1 (1966), 68–74.

[2] Van Nooten, B. (1993-03-01). "Binary numbers in Indian antiquity". *Journal of Indian Philosophy* 21 (1): 31–50. doi:10.1007/BF01092744. http://www.springerlink.com/content/n45g2606g0k76858/ Retrieved 2010-05-06.

[3] http://mkgadiyaram.tripod.com/

[4]"Computing Science in Ancient India" edited by T. R. N. Rao and Subhash Kak

[5] Klaus Mylius, *Geschichte der altindischen Literatur*, Wiesbaden 1983.

[6]Genetic Algorithms and Evolutionary Computation by Adam Marczyk

[7] http://en.wikipedia.org/wiki/Genetic_algorithm

[8] *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman. ISBN 0-7167-1045-5.

[9] The Genetic algorithm in computer science by Eric Krevice Prebys

[10] http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html

[11] Fitness functions in evolutionary robotics: A survey and analysis (AFFG) (PDF), A review of fitness functions used in evolutionary robotics. Retrieved from "http://en.wikipedia.org/wiki/Fitness_function"

[12] http://en.wikipedia.org/wiki/Selection_(genetic_algorithm)

[13 Fogel, David B. (2000). *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*. New York: IEEE Press. pp. 140. (Crossover)

[14] Lilavati by Bhaskarachharya

[14] http://en.wikipedia.org/wiki/Mutation_(genetic_algorithm)

[15] Vrutta Ratnakar

[16] Buthainah Fahran Al-Dulaimi, and Hamza A. Ali Enhanced Traveling Salesman Problem Solving by Genetic Algorithm Technique (TSPGA), World Academy of Science, Engineering and Technology 38 2008

[17] Angel Goñi Moreno, SOLVING TRAVELLING SALESMAN PROBLEM IN A SIMULATION OF GENETIC ALGORITHMS WITH DNA, International Journal "Information Theories & Applications" Vol.15 / 2008