Randomized algorithm approach for solving PCP

Harsh Bhasin
computergrad.com
Faridabad, India
i_harsh_bhasin@yahoo.com

Nishant Gupta
Vellore Institute of Technology, India
Vellore, India
nishantgupta1010@gmail.com

***Abstract*— Post Correspondence Problem is an undecidable problem that was introduced by Emil Post and is often used in proofs of undecidability. No efficient nondeterministic solution to the problem exists. The paper intends to present a nondeterministic solution to the above problem. The proposed work has been tested for some constrained inputs and the results were encouraging. The paper also discusses the application of genetic algorithms to the solution and the requisite analysis. The approach presents an Artificial Intelligence based solution to a problem which is used in theoretical computer science for proving purposes and can be extended to solve many non deterministic problems.**

Keywords- Genetic Algorithms, NP Hard Problem, Artificial Intelligence, Post Correspondence Problem, Non Deterministic Problems.

I. INTRODUCTION

The combinatorial problem known as Post Correspondence Problem (PCP) was first described by Emil L.Post [1]. It was formulated in 1940's. In the most simplistic way, two sets of strings $\{x_1,x_2….x_n\}$ and $\{y_1,y_2….y_n\}$ are taken and a sequence such that $x_{i1}x_{i2}….x_{ik}=y_{i1}y_{i2}…..y_{ik}$ is to be determined. The instance of the problem gives an affirmative answer if two sequences match. The sequences can have repeated patterns as well which makes the problem more difficult.

Genetic algorithms (GAs) are a heuristic search process which is based on theory of genetic modification and survival of fittest. It is applied when search space is too large and the solution can be assigned some fitness value. In PCP there can be many solutions which can be judged by assigning fitness value on the basis of to what extent the strings match. Therefore, PCP is a fit case for applying GAs.

The application of GA to PCP can only make set of solution better. So before applying PCP there must be a randomized algorithm which solves PCP. The work presented proposes a randomized algorithm and then improves it by applying genetic.

II. POST CORRESPONDENCE PROBLEM

As mentioned earlier PCP is an undecidable problem and its undecidability can be theoretically proved [2; 3]. The PCP of 2 pairs has been proved decidable [4], and recently a simpler proof using a similar idea was developed [5].PCP of 7 pairs was proven undecidable [6]. Currently the decidability of PCP of 3 pairs to PCP of 6 pairs is still unknown [7]. The above point makes the case for the work presented.

Undecidability - PCP is undecidable means trying all lists $x_1,x_2….x_k$ in order of k, if we find a solution, the answer is "yes". But if we never find a solution, how can it can be sure that there is no longer solution. So we can never say "no".

PCP can be explained with the help of following example. Given two set of strings are X= {11, 11, 00} and Y= {110, 1, 01}. Here x_0=11, x_1=11, x_2=00 and y_0=110, y_1=1, y_2=01. The sequence x_0 x_2 x_1 makes 110011 and the same sequence is made by y_0 y_2 y_1.

$x_0x_2x_1$ ⇨ 11 00 11⇨ 110011

$y_0y_2y_1$ ⇨ 110 01 1⇨ 110011

Hence it can be seen that the sequence (0, 2, 1) is a solution. Furthermore, since (0, 2, 1) is a solution therefore all of its repetitions; such as (0, 2, 1, 0, 2, 1); are also its solution. There are PCP's that have no solution. For example X= {00, 001, 1000} and Y= {0, 11, 011} are the two strings in which no two sequences match. This is

because total length of strings from Y is smaller than total length of strings from X. It plays a central role in computer science due to its applicability for showing the undecidability of many computational problems in a very natural and simple way [8].

III. NP HARD PROBLEMS

The class P consists of those problems that are solvable in polynomial time. They are problems that can be solved in time $O(n^k)$ for some constant k, where n is the size of the input to the problem [8]
The class NP consists of those problems that are "verifiable" in polynomial time. This means that if we were somehow given a "certificate" of a solution, then we could verify that the certificate is correct in time polynomial in the size of the input to the problem [9].
A problem in P is also in NP, since if a problem is in P then we can solve it in polynomial time without even being given a certificate [9].

P is subset of NP as shown in Fig. 1

A problem is in the class NP-Complete if it is in NP and it is as "hard" as any problem in NP. No polynomial-time algorithm has yet been discovered for an NP-complete problem, nor has anyone yet been able to prove that no polynomial-time algorithm can exists for any one of them [9].
A problem is in class NP-Hard if the problem is “at least as hard as the hardest problems in NP”. A problem H is said to be NP-hard if and only if there is a NP-complete problem L that is polynomial time Turing reducible to H (i.e., $L \leq_T H$).NP-hard problems can be of any type: decision problems, search problems and optimization problems [10]. PCP is an NP-Hard problem.

IV. GENETIC ALGORITHMS

Genetic Algorithms (GAs) helps us to select the optimal solution from a huge set based on some criteria. Generally it is used when search space is too large to apply conventional search methods. It imitates process of genetic evolution. The basic unit of GA is chromosome. A chromosome can be of many types. In the work presented binary chromosome has been used. Binary chromosome represents a set of binary attributes each one called a cell whose value can be either 0 or 1. This combination of cells makes a chromosome. A chromosome can be considered as good or not as good depending upon its fitness value. The fitness value is determined by the transformation that is applied to convert GA into problem at hand.
The initial population generated can be made better by applying genetic operators. They are crossover, mutation and reproduction.

A. Crossover - Crossover operator has the significance as that of crossover in natural genetic process. In this operation two chromosomes are taken and a new is generated by taking some attributes of first chromosome and the rest from second chromosome. In GAs a crossover can be following types

1. *Single point crossover*
   In this crossover, a random number is selected from 1 to n as the crossover point, where n being the number of chromosome. Any two chromosomes are taken and operator is applied as shown in figure [11].

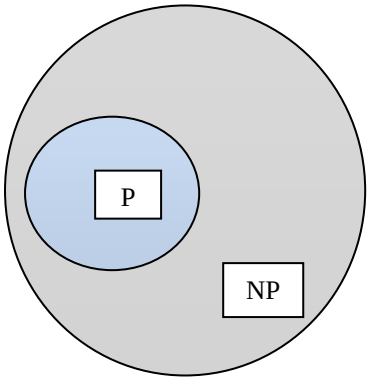

Figure 1. Relation between P and NP problem

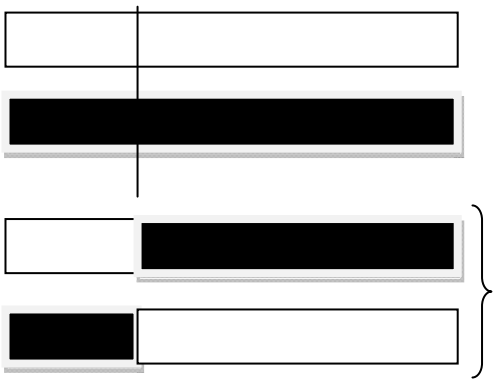

Figure 2. Example of single point crossover

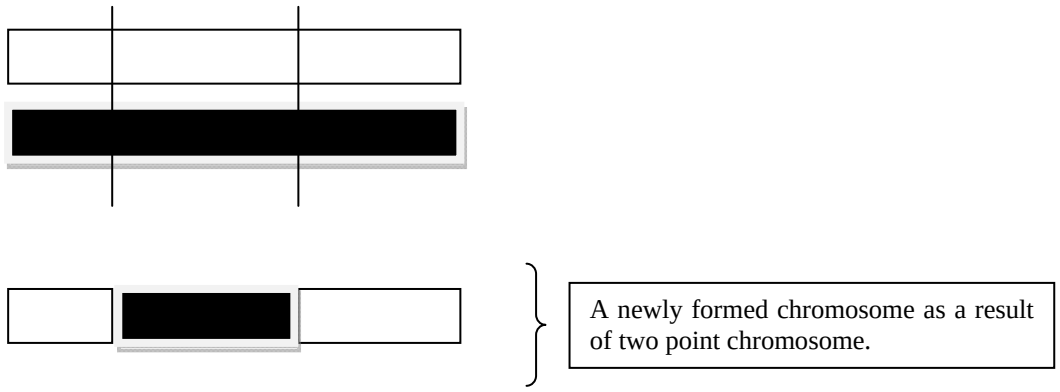

Figure 3. Example of two point crossover

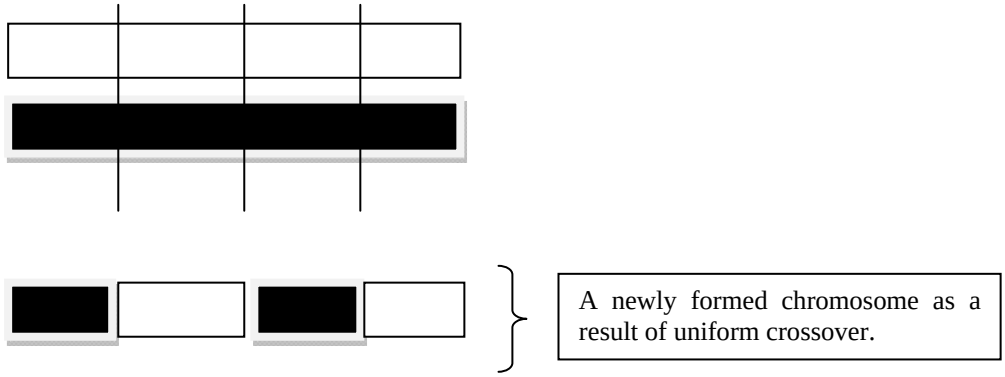

Figure 4. Example of uniform crossover

2. *Two point crossover*
   In this crossover, two crossover points are selected. The resultant is as shown in the Fig. 3.
3. *Uniform crossover*
   In this crossover bits are uniformly copied from both the chromosomes as shown in Fig. 4

B. Mutation – Mutation is genetic operator used to maintain genetic diversity from one generation of population to the next. It is similar to biological mutation [12]. Mutation allows the algorithm to avoid local minima by preventing the population chromosomes from becoming too similar to each other [11].

C. Selection - It is the process of extracting better chromosome from amongst the population. Each chromosome is assigned a fitness value. The chromosome with more fitness value is considered better and should have more probability of being selected. It is done through *Roulette Wheel Selection* [12].

For example four chromosomes have fitness value as shown in TABLE I

Total fitness value = 4+2+3+1=10.
After this, percentage is calculated that they will occupy on roulette wheel by (Fitness value of chromosome * 360) / (Total fitness value).
The Roulette wheel be as shown in Fig. 5
A random number is generated which is equivalent of throwing a dice on the roulette wheel.

TABLE I. FITNESS VALUE OF CHROMOSOMES

CHROMOSOME	FITNESS VALUE
C1	4
C2	2
C3	3
C4	1

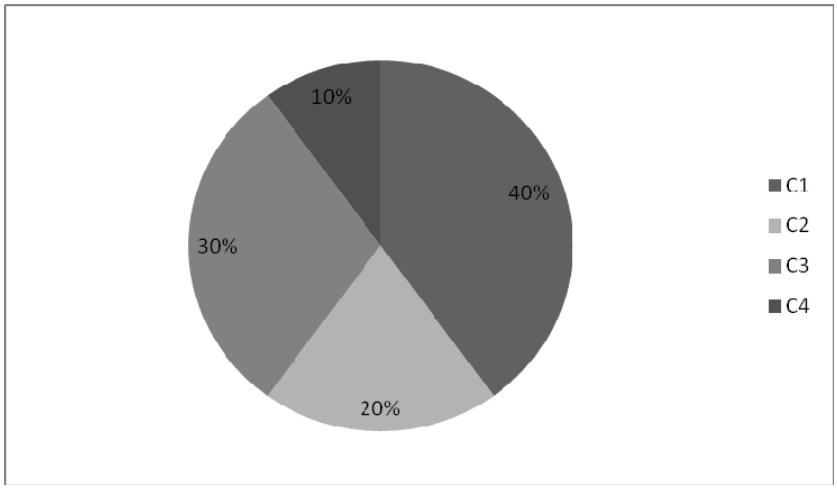

Figure 5. Percentage the chromosomes occupy on pie chart

V. PROPOSED WORK

The following steps describe the process to solve PCP problem by combination of nondeterministic and genetic algorithms.

A. *Population Generation*-Generate the initial binary population of GA. Each chromosome will have 40 cells. Number of chromosome can be entered by the user. The plausibility of getting better result increases if the size of initial population is more (say 40-49).

B. *Cell Division*-Divide each chromosome into group of cells where number of cells in each group are

$$n=x \qquad (1)$$

such that

$$N \leq 2^x \qquad (2)$$

N is the nearest smaller number with respect to power of 2.

C. *Encoding*-Each group now contains n cells. For example n=2, the group of cells will contain either of 00,01,10,11. The above generated groups are converted into decimal form and afterwards, take modulus with respect to N, where N is the number of strings in each set i.e. N_2=decimal (binary form of each group) % N, such that each group N is converted into N_2.

D. *Conversion*-Now each chromosome is converted into 1D array. For example

11 01 10
10 01 00
00 → 0, doing modulo by 3, 0%3=0
01 → 1, doing modulo by 3, 1%3=1
10 → 2, doing modulo by 3, 2%3=2
11 → 3, doing modulo by 3, 3%3=0

This pair of chromosomes is converted into
012
210
This is pair of 1D array in decimal form.

E. *Matching*-After the above step, each part of row of 1D array is matched with every other part of the other rows. Along with this the corresponding sequence of strings is also matched. For example 2 rows of 1D array are 11200211 and 01020210. And the strings are x= {11, 11, 00} and y= {110, 1, 01}. The part of row must match (11200211 and 01020210). Also the corresponding string sequence 110011 is same for both parts (021) of rows. After the same sequence of two rows and strings, solution to the problem is achieved.

F. The above population is subjected to GA operators like crossover and mutation to give the final population.

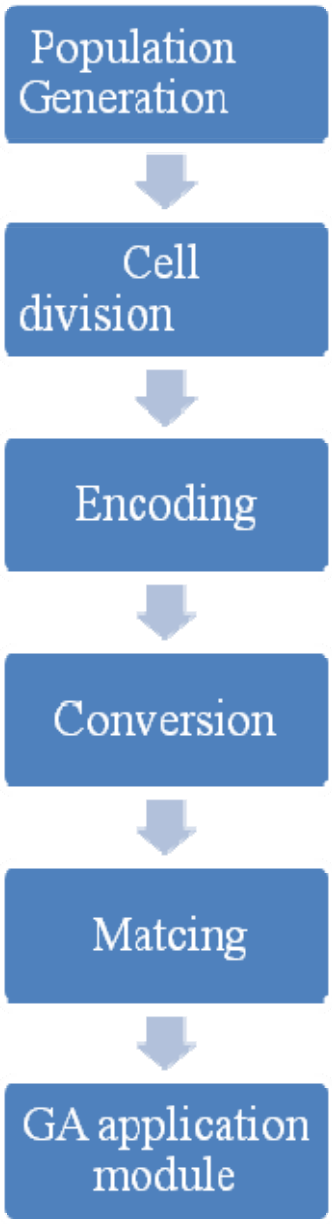

Figure 6. Process based on proposed work

G. *GA application module*-The fitness function of the GA application module will be decided on the basis of deviation of the proceeding strings formed. This gives the value of λ This value of λ will generate fitness value given by $\frac{1}{1+e^{-\lambda}}$

The process has been summarized as shown in Fig. 6

VI. RESULTS AND CONCLUSION

The following is the small part of output which demonstrates that by giving $\{x_1, x_2, x_3\}$ and $\{y_1, y_2, y_3\}$ as input to the program, it produces a result. Last column shows output generated by implementation. Each set input has been executed twice so as to see the effect of non-determinism in algorithm proposed. The results are satisfactory and the program seemingly works for small inputs. The program has been made in recursive enumerable way.

TABLE II. OUTPUT OBTAINED ON RUNNING THE PROGRAM NUMBER OF TIMES ON DIFFERENT INPUTS

	X_1	X_2	X_3	Y_1	Y_2	Y_3	OUTPUT
1.RUN1	11	11	00	110	1	01	110011,110011110011
1.RUN2	11	11	00	110	1	01	110011
2.RUN1	0	01	110	100	00	11	110011100
2.RUN2	0	01	110	100	00	11	110011100
3.RUN1	10	01	11	1	101	011	1011,10111011
3.RUN2	10	01	11	1	101	011	1011,10111011
4.RUN1	11	01	1	1	10	11	111,111111
4.RUN2	11	01	1	1	10	11	11111,111,1101111011
5.RUN1	100	11	111	001	111	11	11111
5.RUN2	100	11	111	001	111	11	11111,11100111111100111
6.RUN1	111	011	111	11	10111	11	111011111
6.RUN2	111	011	111	11	10111	11	111011111

The above implementation has been done in Dot Net framework. The language used is C# and results obtained were analyzed in msxl. The above is just an extract of results obtained. There are certain cases where not so favorable results are found.

The above implementation has been tested and evaluated by plotting two types of graphs. Both the graphs help to evaluate the performance using two distinct parameters. The first graph takes into account the variation of number of same outputs by number of chromosomes and the second graph considers variation of different outputs by number of chromosomes.

Fig. 7 is the graph for number of times the strings matched at particular sequence (here sequence is 2120) when number of chromosomes are varied. It can be said that it is for number of times same sequence is coming by varying number of chromosomes. This is the case where outputs determined by program are {2120 and 21202120}.

For example

String X= {0, 01, 110} Y= {100, 00, 11}.

- If number of chromosomes is 40 then number of times strings matched at sequence 2120 are 10.
- If number of chromosomes is 35 then number of times strings matched at sequence 2120 are 6.

Fig. 8 is the graph for the number of different outputs at single run for given set of strings when numbers of chromosomes are varied. This is the case where outputs determined by program are {210210, 210, 010010, 012, 012010, 212, 010, 012012, and 212210}.

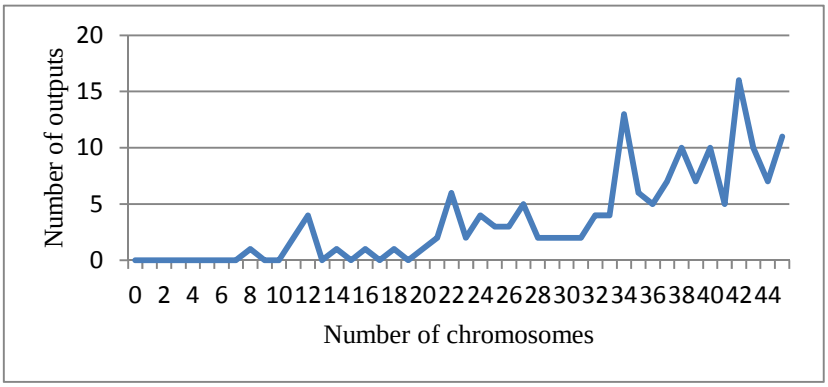

Figure 7. Number of same outputs for number of chromosomes

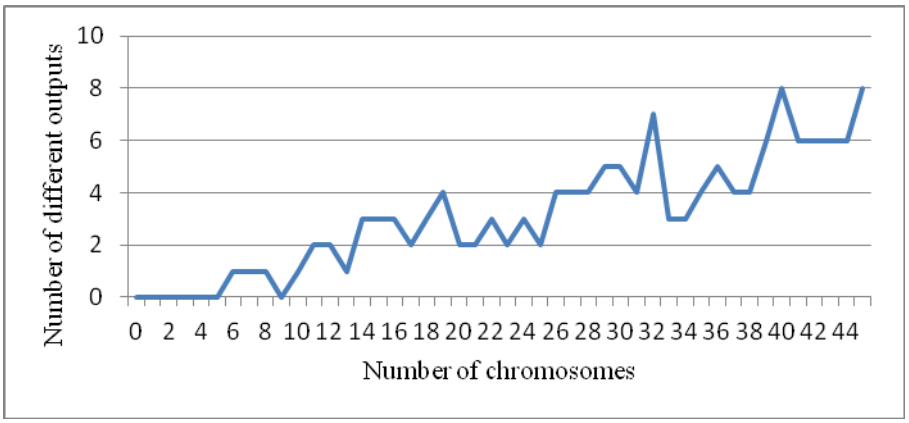

Figure 8. Number of different outputs for number of chromosomes

For example
Strings X= {111, 011, 111} Y= {11, 10111, 11}

- If number of chromosomes is 45 then number of different outputs are 8, that are {210210, 210, 010010, 012, 012010, 212, 010, and 012012}.
- If number of chromosomes is 30 then number of different outputs are 5, that are {012010, 210212, 010010, 012012, and 010}

REFERENCES

[1] E.L.Post. A variant of recursively unsolvable problem, Bulletin of the American Mathematical Society, 52, 264-268, 1946.
[2] Bas van Gijzel, Utrecht University, May 17, 2010.
[3] Peter Linz, "An introduction to formal languages and automata", 2001.
[4] A. Ehrenfeucht, J. Karhumaki and G. Rozenberg. The (generalized) post correspondence with lists consisting of two words is decidable, Theoretical Computing Science, 119-144, 21, 2, 1982.
[5] V. Halava, T. Harju and M. Hirvensalo. Binary (generalized) post correspondence problem, TUCS Technical Report, No. 357, August 2000.
[6] Y. Matiyasevich and G. Senizergues. Decision problem for semi-Thue systems with a few rules, 11th Annual IEEE Symposium on logic in computer science, 1996.
[7] Ling Zhao, Department of computer science, University of Alberta.
[8] P C. Bell and Igor Potapov, "The Identity Correspondence Problem and its Applications", 2010.
[9] T.H. Cormen, C.E. Leiserson, and R.L. Rivest, "Introduction to Algorithms", the MIT Press, Cambridge, Massachusetts, USA, 2009.
[10] Michael R. Garey and David S. Johnson (1979), "Computers and Intractability: A Guide to the Theory of NP-Completeness". W.H. Freeman.
[11] H. Bhasin and S. Bhatia, "Application of Genetic Algorithms in Machine learning", IJCSIT, Vol. 2 (5), 2011.
[12] S. Thrun, "Learning to Play the Game of Chess", In G. Tesauro, D. Touretzky, and T. Leen, editors, Advances in Neural Information Processing Systems(NIPS) 7, Cambridge, MA, 1995. MIT Press.

AUTHORS PROFILE

Harsh Bhasin has completed his B. Tech in Computer Science and Engineering, M. Tech (C. E.). He is presently working in computergrad.com and a member of International Association of Computer Science and Information Technology. He has published many papers in the domain of Random Number generation and NP hard problem by cellular automata; Cryptography, machine learning, equation solving and NP hard problems by

Genetic Algorithm; conversion of south Asian languages by Natural Language Processing. His Various papers have been published in IJCA online, IJCSIT, IJCST and IEEE and National and International conferences.

Nishant Gupta is doing his B. Tech in Computer Science and Engineering from Vellore Institute of Technology, Tamil Nadu, India. He is a member of Computer Society of India (CSI), India.