International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol. 2, Issue. 5, Sep.-Oct. 2012 pp-3275-3280 ISSN: 2249-6645

Crypto-Arithmetic Problem using Parallel Genetic Algorithm (PGA)

Mr. Shedge Kishor Namdeo, Mr. Sravan Kumar G., Miss. Pooja Kuyate
[1](M-Tech Scolar, CSE, RGPV/ LNCT/Indore, Indore, MP.)
[2](Department of Computer, RGPV/ LNCT/ Indore, Indore MP.)
[3](Department of Computer, University of Pune/ SVIT/ PREC, Loni., Nashik, Maharashtra.)

Abstract: *Cryptarithmetic puzzles are quite old and their inventor is not known. An example in The American Agriculturist of 1864 disproves the popular notion that it was invented by Sam Loyd. The name cryptarithmetic was coined by puzzlist Minos (pseudonym of Maurice Vatriquant) in the May 1931 issue of Sphinx, a Belgian magazine of recreational mathematics. In the 1955, J. A. H. Hunter introduced the word "alphabetic" to designate cryptarithms, such as Dudeney's, whose letters from meaningful words or phrases. Solving a cryptarithm by hand usually involves a mix of deductions and exhaustive tests of possibilities. Cryptarithmetic is a puzzle consisting of an arithmetic problem in which the digits have been replaced by letters of the alphabet. The goal is to decipher the letters (i.e. Map them back onto the digits) using the constraints provided by arithmetic and the additional constraint that no two letters can have the same numerical value.*

Keywords: *Genetic algorithms, Parallel Processing, Scheduling, Cryptarithmetic, Parallel Genetic Algorithms.*

I. INTRODUCTION

Cryptarithm is a genre of mathematical puzzle in which the digits are replaced by letters of the alphabet or other symbols. Cryptarithmetic is the science and art of creating and solving cryptarithms. The world's best known Cryptarithmetic puzzle is undoubtedly the puzzle shown in Figure 1. This was first introduced by H.E. Dudeney and was first published in the July 1924 issue of Strand Magazine associated with the story of a Kidnapper's ransom demand [10].

Modernization, by introducing computers and the Internet, is making quite an impact on Cryptarithmetic and it has already become a standard AI problem because it characterizes a number of important problems in computer science arena. A rule based searching technique can provide the solution in minimum time.

```
  S E N D
+ M O R E
-----------
M O N E Y
```

Figure 1: Cryptarithmetic Puzzle

Cryptarithmetic is a class of constraint satisfaction problems which includes making mathematical relations between meaningful words using simple arithmetic operators like 'plus' in a way that the result is conceptually true, and assigning digits to the letters of these words and generating numbers in order to make correct arithmetic operations as well[14].

GENETIC ALGORITHM:

Genetic algorithms were formally introduced in the United States in the 1970s by John Holland at University of Michigan. The continuing price/performance improvement of computational systems has made them attractive for some types of optimization. In particular, genetic algorithms work very well on mixed, combinatorial problems. They are less susceptible to getting 'stuck' at local optima than gradient search methods. But they tend to be computationally expensive. To use a genetic algorithm, you must represent a solution to your problem as a Chromosome. The genetic algorithm then creates a population of solutions and applies genetic operators such as mutation and crossover to evolve the solutions in order to find the best one(s)[4].

MOTIVATION

Cryptarithmetic is a class of constraint satisfaction problems which includes making mathematical relations between meaningful words using simple arithmetic operators like 'plus' in a way that the result is conceptually true, and assigning digits to the letters of these words and generating numbers in order to make correct arithmetic operations as well. A simple way to solve such problems is by depth first search (DFS) algorithm which has a big search space even for quite small problems. I am proposing a solution to this problem with genetic algorithm and then optimized it by using parallelism. I also showed that the algorithm reaches a solution faster and in a smaller number of iterations than similar algorithms.

OBJECTIVES

In the beginning, there are randomly generated individuals. All those individuals create a population. The population in certain time is called a generation. According to their qualities they are chosen by operators for creation of a new generation. The quality of the population grows or decreases and give limits to some constant. Every individual is represented by its chromosome. Mostly chromosomes represented as a binary string. Sometimes there are more strings which are not necessarily of a binary type. The chromosome representation could be evaluated by a fitness function. The fitness equals to the quality of an individual and is an important pick factor for a selection process. The average fitness of a population changes gradually during the run. Operating on the population, several operators are defined. After choosing randomly a pair of individuals, crossover executes an exchange of the substring within the pair with some probability. There are many types of crossovers defined, but a description is beyond the scope of this report. Mutation is an operator for a slight change of one individual/several individual in the population. It is random, so it is against staying in the local minimum. Low mutation parameter means low probability of mutation. Selection identifies the fittest individuals. The higher the fitness, the bigger the probability to

become a parent in the next generation. The computation time for serial GA execution be-comes high for time consuming fitness functions such as those including finite element analysis (FEA) at each objective function call. A better alternative is to take advantage of the intrinsically parallel nature of GAs and to perform the generation of new populations in parallel, on different processors.

II. LITERATURE REVIEW/ SURVEY

Cryptarithmetic is a puzzle consisting of an arithmetic problem in which the digits have been replaced by letters of the alphabet. The goal is to decipher the letters (i.e. Map them back onto the digits) using the constraints provided by arithmetic and the additional constraint that no two letters can have the same numerical value.

Cryptarithmetic is a class of constraint satisfaction problems which includes making mathematical relations between meaningful words using simple arithmetic operators like 'plus' in a way that the result is conceptually true, and assigning digits to the letters of these words and generating numbers in order to make correct arithmetic operations as well[1].

CONSTRAINT SATISFACTION PROBLEM

Cryptarithmetic is a suitable example of Constraint Satisfaction Problem. Instead of providing description, a Cryptarithmetic problem can be better described by some constraints [12].

Constraints of the Cryptarithmetic problem are as follows:

- The arithmetic operations are in decimal; therefore, there must be maximum ten different letters in overall strings which are being used.
- All of the same letters should be bound to a unique digit and no two different letters could be bounded to the same digit.
- As the words will represent numbers, the first letter of them could not be assigned to zero.
- The resulting numbers should satisfy the problem, meaning that the result of the two first numbers (operands) under the specified arithmetic operation (plus operator) should be the third number.

Consider that, the base of the numbers is 10. Then there must be at most 10 unique symbols or letters in the problem. Otherwise, it would not be possible to assign a unique digit to each unique letter or symbol in the problem. To be semantically meaningful, a number must not begin with a zero. So, the letters at the beginning of each number should not correspond to zero.

WHY GENETIC ALGORITHMS?

It is better than conventional AI in that it is more robust. Unlike older AI systems, they do not break easily even if the inputs changed slightly, or in the presence of reasonable noise. Also, in searching a large state-space, multi-modal state-space, or n-dimensional surface, a genetic algorithm may offer significant benefits over more typical search of optimization techniques (linear programming, heuristic, depth-first, breath-first.)[15]. A genetic algorithm (GA) is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are a type of iterative mathematical modeling technique used to find the optimal combinatorial state given a set of parameters of interest.

Genetic algorithms (GAs) are powerful search techniques that are used successfully to solve problems in many different disciplines. Parallel GAs are particularly easy to implement and promise substantial gains in performance and as such there has been extensive research in this field. Genetic algorithms are based on natural selection discovered by Charles Darwin. They employ natural selection of fittest individuals as optimization problem solver. Optimization is performed through natural exchange of genetic material between parents. Offspring's are formed from parent genes. Fitness of offspring's is evaluated. The fittest individuals are allowed to breed only. Offspring's are created during crossover and mutation. The crossover is an operation when new Chromosomes offspring's are produced by fusing parts of other chromosomes parents. The mutation is random replacement of chromosome bits. Thus, offspring's form a new generation which replaces the old one.

The success of optimization strongly depends on the chosen chromosome encoding scheme, crossover and mutation strategies as well as fitness function. For each problem, careful analysis must be done and correct approach chosen. As it was shown, one Chromosome can contain a whole image or only a small part of it, a whole parameter range or only the most descriptive ones. Crossover can be performed in various manners, for example by exchanging information at one brake point or at several one.

III. PARALLEL GENETIC ALGORITHMS

If we mimic natural evolution we would not operate on a single population in which a given individual has the potential to mate with any other partner in the entire population Instead, species would tend to reproduce within subgroups or within neighborhoods. A large population distributed among a number of semi-isolated breeding groups is known as polytypic. A PGA introduces the concept of interconnected demes. The local selection and reproduction rules allow the species to evolve locally, and diversity is enhanced by migrations of strings among demes [13].

In a genetic algorithm, a population of strings (called chromosomes or the genotype of the genome), which encode candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem, evolves toward better solutions. Traditionally, solutions are represented in binary as strings of 0s and 1s, 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, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has 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. Genetic algorithms find application in bioinformatics, computational science, engineering, economics, chemistry, manufacturing, mathematics, physics and other fields[4].

A TYPICAL GENETIC ALGORITHM REQUIRES:
A genetic representation of the solution domain,
A fitness function to evaluate the solution domain.

A standard representation of the solution is as an array of bits. Arrays of other types and structures can be used in essentially the same way. The main property that makes these genetic representations convenient is that their parts are easily aligned due to their fixed size, which facilitates simple crossover operations. Variable length representations may also be used, but crossover implementation is more complex in this case. Tree like representations are explored in genetic programming and graph form representations are explored in evolutionary programming. Evolutionary programming (EP) involves populations of solutions with primarily mutation and selection and arbitrary representations. They use self adaptation to adjust parameters, and can include other variation operations such as combining information from multiple parents.

The fitness function is defined over the genetic representation and measures the quality of the represented solution. The fitness function is always problem dependent. For instance, in the knapsack problem one wants to maximize the total value of objects that can be put in a knapsack of some fixed capacity. A representation of a solution might be an array of bits, where each bit represents a different object, and the value of the bit (0 or 1) represents whether or not the object is in the knapsack. Not every such representation is valid, as the size of objects may exceed the capacity of the knapsack. The fitness of the solution is the sum of values of all objects in the knapsack if the representation is valid or 0 otherwise. In some problems, it is hard or even impossible to define the fitness expression; in these cases, interactive genetic algorithms are used.

Once the genetic representation and the fitness functions are defined, GA proceeds to initialize a population of solutions randomly, and then improve it through repetitive application of mutation, crossover, inversion and selection operators.

Initialization

Initially many individual solutions are randomly generated to form an initial population. The population size depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions. Traditionally, the population is generated randomly, covering the entire range of possible solutions (the search space). Occasionally, the solutions may be "seeded" in areas where optimal solutions are likely to be found.

Selection (genetic algorithm)

During each successive generation, a proportion of the existing population is selected to breed a new generation. Individual solutions are selected through a fitness based process, where fitter solutions (as measured by a fitness function) are typically more likely to be selected. Certain selection methods rate the fitness of each solution and preferentially select the best solutions. Other methods rate only a random sample of the population, as this process may be very time consuming.

Most functions are stochastic and designed so that a small proportion of less fit solutions are selected. This helps keep the diversity of the population large, preventing premature convergence on poor solutions. Popular and well studied selection methods include roulette wheel selection and tournament selection.

Reproduction

Crossover (genetic algorithm) and Mutation (genetic algorithm)

The next step is to generate a second generation population of solutions from those selected through genetic operators: crossover (also called recombination), and/or mutation. For each new solution to be produced, a pair of "parent" solutions is selected for breeding from the pool selected previously. By producing a "child" solution using the above methods of crossover and mutation, a new solution is created which typically shares many of the characteristics of its "parents". New parents are selected for each new child, and the process continues until a new population of solutions of appropriate size is generated. Although reproduction methods that are based on the use of two parents are more "biology inspired", some research suggests more than two "parents" are better to be used to reproduce a good quality chromosome.

These processes ultimately result in the next generation population of chromosomes that is different from the initial generation. Generally the average fitness will have increased by this procedure for the population, since only the best organisms from the first generation are selected for breeding, along with a small proportion of less fit solutions, for reasons already mentioned above. Although, Crossover and Mutation are known as the main genetic operators, it is possible to use other operators such as regrouping or migration in genetic algorithms.

- Simple generational genetic algorithm pseudo code:
- Choose the initial population of individuals.
- Evaluate the fitness of each individual in that population.
- Repeat on this generation until termination: (time limit, sufficient fitness achieved, etc.)
- Select the best-fit individuals for reproduction .
- Breed new individuals through crossover and mutation operations to give birth to offspring.
- Evaluate the individual fitness of new individuals.
- Replace least-fit population with new individuals.

Genetic algorithms with adaptive parameters (adaptive genetic algorithms, AGAs) is another significant and promising variant of genetic algorithms. The probabilities of crossover (pc) and mutation (pm) greatly determine the degree of solution accuracy and the convergence speed that genetic algorithms can obtain. Instead of using fixed values of pc and pm, AGAs utilize the population information in each generation and adaptively adjust the pc and pm in order to maintain the population diversity as well as to sustain the convergence capacity. In AGA (adaptive genetic algorithm), the adjustment of pc and pm depends on the fitness values of the solutions. In CAGA (clustering based adaptive genetic algorithm), through the use of clustering analysis to judge the optimization states of the population, the adjustment of pc and pm depends on these optimization states. It can be quite effective to combine GA with other optimization methods. GA tends to be quite good at finding generally good global solutions, but quite inefficient at finding the last few mutations to find the absolute optimum. Other techniques (such as simple hill climbing) are quite efficient at finding absolute optimum in a limited region. Alternating GA and hill climbing can

improve the efficiency of GA while overcoming the lack of robustness of hill climbing [6].

Genetic operators as independent parts of GA

The parallel steady-state genetic algorithm with tournament bad individual selection was implemented. In this implementation [6] the genetic algorithm consists of two threads: one performs tournament selection and crossover and the other mutation. The major problem of that simple parallel implementation is that it has no control over mutation probability. The consequence is a very bad algorithm behavior. The results are slightly better than random search, but also useless. If the threads are left to parallel execution without any control, one of two threads can waste some time on waiting for processor time.

An Evolutionary Algorithm will search for solutions in shortest time but the performance will also reflect the toughness of the problem. A parallel genetic algorithm has been developed to dynamically schedule heterogeneous tasks to heterogeneous processors in a distributed environment. The proposed algorithm uses multiple processors with centralized control for scheduling. Tasks are taken as batches and are scheduled to minimize the execution time and balance the loads of the processors.

IV. SUMMARY & DISCUSSION

In this project we try to analyze an efficient Parallel genetic algorithm to solve Cryptarithmetic Problems. Additionally, it illustrates how to plug in techniques of Evolutionary Approach into Constraint Satisfaction Problem. This sort of design can provide efficient solution to a wide range of Constraint Satisfaction Problem or other generic searching problems that could be characterized as a Constraint Satisfaction Problem as well. This parallel model has been tested in order to determine the best method for comparing, science it uses two platform-independent parameters; the number iteration and java programming language. So, further research should go on to optimize the main proposed parallel ideas in the near future. This project concentrated on solving Cryptarithmetic problems in an efficient way. The use of parallel genetic algorithm showed that we can even find the result of large instances of this problem within an acceptable time.

Discussion is related a simple Cryptarithmetic problem solution in stepwise mode - Cryptarithmetic is a CSP problem in which letters are substituted by digits such that each letter represents a unique digit, and the actual problem is to find a proper sequence of digits assigned to different letters satisfying the conditions of the arithmetic operation. What is a Cryptarithmetic problem? It is a mathematical puzzle in which each letter represents a digit (for example, if X=3, then XX=33). The object is to find the value of each letter. No two letters represent the same digit (If X=3, Y cannot be 3). And the first letter cannot be 0 (Given the value ZW, Z cannot be 0). They can be quite challenging, often involving many steps. Here's an example, illustrating how to solve them:

```
  SEND
+ MORE
-------
 MONEY
```

M must be 1. This is an addition problem; the sum of two four digit numbers can't be more than 10,000, and M can't be 0 according to the rules since it's the first letter. So now we have:

```
  SEND
+ 1ORE
-------
 1ONEY
```

Now in the column S1O, S+1≥10. S must be 8 (if there is a 1 carried over from the column E0N) or 9. O must be 0 (if S=8 and there is a 1 carried or S=9 and there is no 1 carried) or 1 (if S=9 and there is a 1 carried). But 1 is already taken, so O must be 0.

```
  SEND
+  10RE
-------
 10NEY
```

There can't be a carry from the column E0N, because any digit plus 0 < 10, unless there is a carry from the column NRE and E=9; but this cannot be the case, because then N would be 0, and 0 is already taken. So E<9 and there is no carry from this column. Therefore, S=9, because 9+1=10.

In the column E0N, E cannot be equal to N, so there must be a carry from the column NRE; E+1=N. We now look at the column NRE; we know that E+1=N. Since we know that there is a carry from this column, N+R=1E (if there is no carry from the column DEY) or N+R+1=1E (if there is a carry from the column DEY). Let's try out both cases. No carry: N+R=10+(N-1)=N+9,

R=9; 9 is already taken, so this won't work.

Carry: N+R+1=N+9; R=8. This must be the solution for R.

```
  9END
+ 108E
-------
 10NEY
```

The digits we have left are 7, 6, 5, 4, 3, and 2. We know there must be a carry from the column DEY, so D+E>10. N=E+1, so E can't be 7 because then N would be 8 which is already taken. D is at most 7, so E cannot be 2 because then D+E<10, and E cannot be 3 because then D+E=10 and Y=0, but 0 is taken already. Likewise, E cannot be 4 because if D>6, D+E<10, and if D=6 or D=7, then Y=0 or Y=1, which are both taken. So E is 5 or 6. If E=6, then D=7 and Y=3, so this part works. But look at the column N8E. Remember, there is a carry from the column D5Y. N+8+1=16 (because we know there is a carry for this column). But then N=7, and 7 is taken by D. Therefore, E=5.

```
  95ND
+ 1085
-------
 10N5Y
```

Now that we've gotten this important digit, it gets much simpler from here. N+8+1=15, N=6.

```
  956D
+ 1085
------
 1065Y
```

The digits left are 7, 4, 3, and 2. We know there is a carry from the column D5Y, so the only pair that fits is D=7 and Y=2

```
  9567
+ 1085
------
 10652
```

The problem is solved! These are quite tricky and require some thinking, but are lots of fun. Now we'll take turns posting problems. When a problem is solved.

V. CONCLUSION

In Cryptarithmetic puzzle, the arithmetic operations are simple and of base 10, but are ciphered into letters. The task is to decipher them. Here we concentrated on solving Cryptarithmetic problems in an efficient way. Parallel implementations of Genetic Algorithms are very performable to solve large scale problems. The use of parallel genetic algorithm showed that we can even find the result of large instances of this problem within an acceptable time. The proposed algorithm uses multiple processors with centralized control for scheduling. Tasks are taken as batches and are scheduled to minimize the execution time and balance the load among of the processors. A scheduling algorithm has been developed to schedule heterogeneous tasks onto heterogeneous processors on a distributed environment.

Genetic Algorithms are powerful but usually suffer from longer scheduling time which is reduced in our algorithm due to the parallelization of the fitness evaluation. The proposed algorithm uses a straightforward encoding scheme and generates a randomized initial population. The fitness function uses the maxspan, balance of load among the processors and communication costs while evaluating the schedules. By parallelization I got a better program structure and a significant decrease in computational time on a multiprocessor system. As per the implementation, testing, result analysis I conclude that PGA and DFS implementation is 80 to 90 % successful.

REFERENCES

[1]. Abu Sayef Md. Ishaque, Md. Bahlul Haider, Muhammad Al Mahmud Wasid, Shah Mohammed Alaul, Md. Kamrul Hassan, Tanveer Ahsan, Md. Shamsul Alam: "An Evolutionary Algorithm to Solve Cryptarithmetic Problem", International Conference on Computational Intelligence 2004: 494-496.

[2]. AJ Page, TJ Naughton "Dynamic Task Scheduling using Genetic Algorithms for Heterogeneous Distributed Computing", - Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05)

[3]. A. Y. Zomaya, M. Clements, and S. Olariu. "A framework for Reinforcement-Based Scheduling in Parallel Processor Systems", IEEE Transactions on Parallel and Distributed Systems, 9(3):249 260, March 1998.

[4]. Alippi, C., Filho, J.L.R., Treleaven, P.C. (1994), "Genetic-Algorithm Programming Environments", IEEE Trans. Computer, June 1994.

[5]. Bonnie Averbach and Orin Chein, Problem Solving Through Recreational Mathematics, Courier Dover Publications, 2000, pp. 156.

[6]. Budin, L., Golub, M., Jakobovic, D., Parallel Adaptive Genetic Algorithm, International ICSC/IFAC Symposium on Neural Computation NC'98, Vienna, 1998, pp. 157-163.

[7]. Cantú-Paz, E., "Designing Efficient Master Slave Parallel Genetic Algorithms, Genetic Programming": Proceedings of the Third Annual Conference. (pp. 455). San Francisco, CA, 1998.

[8]. Cantú-Paz E., A Summary of Research on Parallel Genetic Algorithms, 1995., available from: www.dai.ed.ac.uk/groups/evalg/Local_Copies_of_Papers/Cantu-Paz.A_Summary_of_Research_on_Parallel_Genetic_Algorithms.ps.gz

[9]. Cantú-Paz, E., A Survey of Parallel Genetic Algorithms, Calculateurs Paralleles, Vol. 10, No. 2. Paris: Hermes, 1998., available via ftp from: ftp://ftp-illigal.ge.uiuc.edu/pub/papers/Publications/cantupaz/survey.ps.Z.

[10]. Cantú-Paz, E., Designing Efficient Masterslave Parallel Genetic Algorithms, Genetic Programming: Proceedings of the Third Annual Conference. (pp. 455). San Francisco, CA, 1998.

[11]. Cantú-Paz, E., Goldberg, D.E., Parallel Genetic Algorithms with Distributed Panmictic Populations, 1999. available from: http://wwwilligal.ge.uiuc.edu/cgi-bin/orderform/orderform.cgi.

[12]. Cantú-Paz, E., Migration Policies, Selection Pressure, and Parallel Evolutionary Algorithms,1999.,available via ftp from: ftp://ftpilligal.ge.uiuc.edu/pub/papers/IlliGALs/99015.ps.Z.

[13]. David Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.

[14]. E. Hou, N. Ansari, and H. Ren. "A Genetic Algorithm for Multiprocessor Scheduling". IEEE Transactions on Parallel and Distributed Systems, 5(2):113–120, February 1994.

[15]. E Cantu-Paz ,A survey of parallel genetic algorithms, Calculateurs Paralleles, Reseaux et Systems Repartis, 1998

[16]. Goodman, E.D., Averill, R.C., Punch, W.F., Eby, D.J., Parallel Genetic Algorithms in the Optimization of Composite Structures, Second World Conference on SoftComputing (WSC2), June, 1997., available from: http://garage.cps.msu.edu/papers/GARAGe97-05-02.ps

[17]. H. E. Dudeney, in Strand Magazine vol. 68 (July 1924), pp. 97 and 214.

[18]. M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund. "Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems." Journal of Parallel and Distributed Computing, 59(2):107– 131, November 1999.

[19]. Michalewicz, Z. (1992), Genetic Algorithms + Data Structures = Evolutionary Programs, Springer-Verlag, Berlin.

[20]. R. Gholamali , D. Gholamhossein :"An Efficient Parallel Algorithm for Solving Cryptarithmetic Problems: PGA", Third UKSim European Symposium on Computer Modeling and Simulation 2009.

[21]. R.Nedunchelian, K.Koushik, N.Meiyappan, V.Raghu, "Dynamic Task Scheduling Using Parallel Genetic Algorithms for Heterogeneous Distributed Computing", Proceedings of the 2006 International Conference on Grid, Las Vegas, Nevada, USA, 2006.

[22]. R. Abbasian, M. Mazloom : "Solving Cryptarithmetic Problems Using Parallel Genetic Algorithm" 2009 Second International Conference on Computer Engineering 978-0-7695-3925-6/09 2009 IEEE

[23]. Stuart Russell, Artificial Intelligence A Modern Approach, Pearson Education, Inc, ISBN 0-13-790395-2, 2003.

[24]. SunSoft, (1994), Solaris 2.4: Multithreaded Programming Guide, Sun Microsystems, Mountain View, California

[25]. Manisha Neaghare, Solving Verbal Arithmetic Problem by Efficient Evolutionary Algorithm" International Conference on Sunrise Technology iCOST2011-Computer Engg. Organised by SSVPS Engineering College, Dhule.

[26]. Manisha Neaghare "Comparison of Parallel Genetic Algorithm with Depth First Search Algorithm for Solving Verbal Arithmetic Problems" International Conference on Emerging Trends in Technology ICWET2011 Organised by Thakur Engineering College, Mumbai published on Association for Computing Machinery ACM SIGART USA and International Journal of Computer Application.

[27]. Shedge Kishor N., Ashwini Verma, Gade Shyam A. "Solving Verbal Crypto-Arithmetic Problem by Parallel Genetic Algorithm (PGA)" International Journal of Computer Technology and Electronics Engineering (IJCTEE) Volume 2, Issue 4, August 2012