International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 05 Issue: 03 | March-2021 ISSN: 2582-3930

Cryptarithmetic Algorithm and Parallel Genetic Algorithm Using for Constraint Satisfaction Problems

[1]ZinMinnThant, [2]SiSiSann, [3]YiYiWin, [4]KhaingZarMyintAung
[1]Faculty of Computer Science, University of Computer Studies (Kalay)
[2]Faculty of InformationTechnology Support and Maintenance, University of Computer Studies (Kalay)
[3]Faculty of Computer Science,University of Computer Studies (Pakokku)
[4] Computer Department, Myanmar Aerospace Engineering University(Meikhtilar)

Abstract -*Constraint Satisfaction Problem is considered as an importantarea of the Mathematical Computation. Constraints are a set of conditions that must be true for some problems like cryptarithmetic problems. Cryptarithmetic problems are where numbers are changed with alphabetic or symbolic order. A popular problem in the field of Artificial Intelligence and the approach used in our paper is cryptarithmetic problems. We suggested a solution to this Genetic Algorithm problem in this paper and then improved it by using parallelism. This algorithm will find the solution using less memory more easily.*

Key Words:Artificial Intelligence, Constraint Satisfaction, Cryptarithmetic Problem, Parallel Genetic Algorithm, Depth-First Search.

1.INTRODUCTION

A puzzle consisting of an arithmetic problem in which the digits have been replaced by letters of the alphabet is Cryptarithm or Word Addition. The objective is to use the constraints to decode the letters (i.e., map them back to the digits) provided that no two letters can have the same numerical value[1].In the May 1931 issue of Sphinx, a Belgian recreational mathematics magazine, this form of problem was popularized and was translated by Maurice Kraitchik in 1942 as "Cryptarithmetic." Fig. 1 demonstrates one of the well-known Cryptarithm or Word Addition.[2]

```
  Y O U R
+ Y O U
---------
H E A R T
```

Figure 1. Cryptarithm problem example

For each message, the consumer must choose a numerical value such that the equation holds a good value. A reasonable solution which is arithmetically accurate would be to assign digits to letters in the following way. E=0, H=1, U=2, A=3, O=4, R=6, T=8, and Y=9 indicate that this means thatFigure 2. An acceptable solution to problem in Fig 1.

```
  9 4 2 6
+ 9 4 2
---------
1 0 3 6 8
```

Figure 2. An acceptable solution to the Cryptarithmetic problem

2. RELATED WORK

In this part, three parallel genetic algorithm applications are used as examples of work related to the current paper.

In the first application of parallel processing systems, the maximization of system performance by task mapping is a fundamental consideration.

An effective allocation strategy will enhance the use of resources and significantly improve the system's throughput.In order to meet performance criteria, such as minimizing execution time or communication delays, we show how to map the tasks between the processors. We rely on Local Neighborhood Search (LNS) as this approach has been shown to outperform a large number of heuristic-based algorithms. We call our LNS based mapping algorithm, Genetic Local Neighborhood Search (P-GLNS). All three of these map-ping strategies were implemented and compared. The experimental results show that 1) the GLNS algorithm is better than the LNS algorithm and 2). Near linear speedup is achieved by the P-GLNS algorithm [4]

In the second application, a parallel genetic algorithm was developed to dynamically schedule heterogeneous tasks in a distributed environment for heterogeneous processors.It is known that the scheduling problem is NP complete. In this field, genetic algorithms, a meta-heuristic search technique, have been successfully used.The proposed algorithm uses multiple processors for scheduling with centralized control. Tasks are taken as batches and are planned to minimize the time of execution and balance the processors' loads.The proposed parallel genetic algorithm (PPGA) substantially reduces scheduling time without adversely affecting the max span of the resulting schedules, according to our experimental findings.[3]

In the third application an investigation is done in embedded systems. Therefore, energy efficiency becomes one of the major design concerns for embedded systems. The technique of dynamic voltage scaling (DVS) can be exploited to reduce the power consumption of modern processors by slowing down the processor speed. The problem of static DVS scheduling in distributed systems such as the energy consumption of the processors is minimized while guaranteeing the timing constraints of the tasks is an NP hard problem. This paper describes a Parallel Genetic Algorithm (PGA) that improves over Genetic Algorithm (GA) for finding better schedules with less time by parallelizing the GA algorithms to run on a cluster [5].

3. CONSTRAINT SATISFACTION PROBLEM

Constraints of Cryptarithmetic problem are as follows:

1. The arithmetic operations are in decimal; therefore, there must be maximum ten different letters in overall strings which are being used.
2. All the same letters should be bound to a unique digit and no two different letters could be bounded to the same digit.
3. As the words will represent numbers, the first letter of them could not be assigned to zero.
4. 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.[8]

Search algorithms inspired by genetics and natural selection are Genetic Algorithms (GAs). These algorithms are effective search techniques that are used in many disciplines to solve difficult issues. Unfortunately, in terms of processing load and memory, they can be very demanding. Parallel Genetic Algorithms (PGAs) are parallel gas implementations that can deliver major performance and scalability gains. The most significant advantage of PGAs is that, even though parallelism is simulated on traditional machines, they have better performance in certain cases than single population-based algorithms. In order to solve decimal Cryptarithmetic problems, this paper proposes an effective parallel genetic algorithm and contrasts the proposed algorithm with ordinary ways to solve them.[9]

4.BASIC OF GENETIC ALGORITHMS

Randomly generated individuals are present in the beginning. All those individuals create a population. Sometimes, the population is referred to as a generation. They are chosen by the operators according to their characteristics for the output of a new generation. The quality of the population increases or decreases and some constant limits are given. A chromosome represents each individual. As binary strings, chromosomes are also represented. Sometimes there are more strings which are not necessarily of a binary nature.

The chromosome representation could be evaluated by a fitness function. Fitness is equal to the quality of a individual and is an important factor in the process of selection. During the run, the average fitness of a population improves gradually.Several operators are characterized by working on the population.*Crossover* performs an exchange of the substring within the pair with some probability after randomly selecting a pair of people. Many types of crossovers are specified, but there is a definition beyond the scope of this paper.

Mutation is an operator in the population for a small shift in one individual /several individual. It is random, so it's against remaining at the minimum local level.

A low parameter for mutation means a low likelihood of mutation. Option distinguishes individuals with fitness. The higher the fitness, the greater the probability of being a parent in the next generation. There are various patterns of selection, but the basic functionality is the same.[10]

5. FORMULATION OF THE ALGORITHM

A. Encoding Individuals

The first part of using GA to solve a problem is to identify a way to encode chromosomes for individuals. Suppose we are dealing with decimal numbers, then we can use a 10-length sequence of characters with indices 0 through 9; if we need to allocate a letter to a digit, we just put that letter to a letter, we just put that letter to the cell that has the array index. For example, in Fig. 3 we have assigned 3 to H.

We should leave unassigned columns of the array empty and to show that they are empty cells we marked it with "- "letter as shown in Fig. 3.

A R T H E Y O U - -

0 1 2 3 4 5 6 7 8 9

Figure 3. A sample chromosome

Since each chromosome is a solution to the problem, it may lead us to another solution by chaining the position of the letters with each other or with empty cells. Test out Fig. 4.

B. Fitness Function

Fitness function evaluates the quality of an individua. The fitness function for this problem is defined with the following formula:

$$FITNESS = |R - (F + S)|$$

Assuming R as the result of the operation, F as the first operand and S as the second operand.

We will get better individuals as the health of individuals gets closer to zero. If an individual's fitness is exactly zero, then that person is the correct solution.

```
 Y O U R
 5 6 7 1
+Y O U
   5 6 7
---------
H E A R T
3 4 0 1 2
```

$$FITNESS = |34012 - (5671 + 567)| = 27774$$

Figure 4. shows the fitness of the chromosome

C. Mutation

Randomly, the mutation operator produces two numbers between 0 and 9 and exchanges the cell content of these two chromosome indices. The mutation operator should ensure that the exchange is not illegal, which means that an exchange that might result in any starting letter of the words being inserted in the chromosome's zero index should not be allowed.

(a) Y O UH - EA - T R
0 1 2 3 4 5 6 7 8 9

(b) Y O U – H E – A T R

0 1 2 3 4 5 6 7 8 9

(c) Y O – U H E – A T R

0 1 2 3 4 5 6 7 8 9

Figure 5. Three chromosomes after mutation

Here, the only restriction is that the letter should not be the starting letter of the chromosome at the start of a number in the problem. On the topic of Fig. 1. S and M cannot be 0. This is because they are the starting numbers in the problem. Furthermore, a more efficient mutation operator should not allow the exchange of empty cells.[6]

D. Making new Generations

In this paper, we have used asexual genetic algorithms; thus, we do not have a crossover of two parents, and the method of generating new generations is simply to use mutation operations on selected individuals.[7]

If we take more chromosomes, more computation will be required, and if we take less chromosomes, we will not be able to reach the solution in a limited time. We'll find the solution after generating several generations when the error in the chromosome is 0. The chromosome solution from Fig. 1 is shown in Fig. 6.

E H U A O – R – T Y
0 1 2 3 4 5 6 7 8 9

Y O U R
9 4 2 6
+ Y O U
9 4 2

H E A R T
1 0 3 6 8

$FITNESS = |10368 - (9426 + 942)| = 0$

Figure 6. The Solution Chromosome

E. Parallelization

Parallelization is applied to the new generations' process of creation. There is a coordinator thread responsible for organizing, identifying and distributing the most relevant individuals and some generator threads responsible for producing new generations. There is an internal population of each thread.Each thread generates a random population of a fixed size at the beginning and includes them in its internal population.

As the thread generates a new individual, it in the right place of its internal population by using insertion sort in an increasingly manner according to individuals' fitness. This way always the better individuals have lower indices and if there is a solution, it would be in index 0.

When all of the threads generated new population, the coordinator thread picks a number of best individuals plus some randomly chosen individuals from each thread's internal population and accumulates them in a list and then distributes these chosen individuals over all the threads as the individuals that the threads use to generate a new population.

During the process of selecting the best people from the internal population of each thread, this process continues until the coordinator thread finds an individual with a fitness equal to zero.

6. PROPOSED ALGORITHM

A. Main Algorithm

1. Put the coordinator thread in wait state.
2. Initialize N generator threads with an internal population of size P.
3. In each generator thread do the following:
   - 3.1 If the inner population is empty generate a random population then go to 3.3, else go to
   - 3.2 Wait for a synchronization signal from coordinator thread and once it has been received, produce new children from the list of individuals which the coordinator has passed.
   - 3.3 Notify the coordinator that the current job is done and ready. Go to step 3.2.
4. Wait for all generator threads to notify the coordinator threads.
5. In coordinator thread, search the inner populations' first indices for the correct solution. If the solution has been found finish the algorithm else go to 6.
6. In coordinator thread, pick R best and some randomly chosen individuals from all internal populations of threads and create a list with them.
7. Distribute the chosen population to the generator threads, signal them, put coordinator thread in wait state and go to step 4.

B. Individual with Fitness Algorithm

1. Extract distinct letters from input strings and put them in a list name L.
2. Repeat the following until the desired population size is reached:
   - 2.1 For each letter in L do the following:
     - 2.1.1 Generate a random number between 0 and 9.
     - 2.1.2 If the random number is equal to zero and the letter is the beginning letter of the words (which should not be zero) go to 2.1.4.
     - 2.1.3 If the cell that is corresponding the random number is empty, put the letter into that cell, else go to step 2.1.1.
     - 2.1.4 If the current letter is the last letter of L and the length of L is 10 then generate a random number between 1 and 9 exchange the place of the letter from that cell to the index of zero and the current letter to that position. Keep doing it until the second letter is not a beginning letter of the words. Else go to step 2.1.1.
   - 2.2 Calculate the fitness of this new individual.
   - 2.3 Add the individual into the right place in the inner population list.

C. Making New Generation Algorithm

1. Get the chosen parent and generate two random numbers.
2. If the two random numbers are not the same or not indicating two empty cells in the parent and will not

lead us to numbers beginning with zeros, exchanges the content of the cells corresponding by the random numbers. Else go to step 1.

3. Calculate the fitness of the new individual, put it in the right place in the inner population and remove the worst individual from the population.

7. EXPERIMENTAL RESULTS

This project was effective in solving cryptarithmetic problems. The use of the parallel genetic algorithm and depth-first search showed that within an acceptable one, we can even find the result of large instances of this problem.Each issue has been checked a hundred times and the average time has been gathered. Table 1 through 5 displays in detail the outcomes.

TABLE 1. RESULT OF A FIVE VARIABLE PROBLEM

Problem (per 100 runs)	DAN + NAN = NORA
Answer	921+121 =1042
Variable	5
Minimum Time (ms)	16
Maximum Time (ms)	172
Average Time (ms)	62

5 generator threads are used in the problem outlined in Table 1.

TABLE 2. RESULT OF A SIX VARIABLE PROBLEM

Problem (per 100 runs)	TWO +TWO = FOUR
Answer	836 +836=1672
Variable	6
Minimum Time (ms)	57
Maximum Time (ms)	224
Average Time (ms)	148

12 generator threads are used in the problem outlined in Table 2.

TABLE 3. RESULT OF A SEVEN VARIABLE PROBLEM

Problem (per 100 runs)	BASE+BALL =GAMES
Answer	7483+7455 =14938
Variable	7
Minimum Time (ms)	230
Maximum Time (ms)	960
Average Time (ms)	506

14 generator threads are included in the problem outlined in Table 3.

TABLE 4. RESULT OF A EIGHT VARIABLE PROBLEM

Problem (per 100 runs)	CAT+RUN =AWAY
Answer	517+ 693 =1210
Variable	8
Minimum Time (ms)	510
Maximum Time (ms)	2130
Average Time (ms)	1155

5 generator threads were used in the problem outlined in Table 4.

TABLE 5. RESULT OF A TEN VARIABLE PROBLEM

Problem (per 100 runs)	BROWN+YELLOW=PURPLE
Answer	52813+649981=702794
Variable	10
Minimum Time (ms)	430
Maximum Time (ms)	3632
Average Time (ms)	2421

25 generator threads are included in the problem outlined in Table 5.

Another experiment was to calculate the average time for problem sizes to solve Cryptarithmetic. Different instances of various variable numbers have been resolved hundreds of times and the average time has been calculated for each problem size. The results of this experiment are shown in Table 6.

TABLE 6. THE RESULT OF DIFFERENT PROBLEMS SIZES

Variable Number	Average Time(ms)
5	112
6	325
7	877
8	1155
10	3511

Fig 7. Illustrate the comparison between the proposed algorithm (PGA) and the depth-first search algorithm (DFS). It is clear from Fig 7. That the PGA has far better results than DFS.

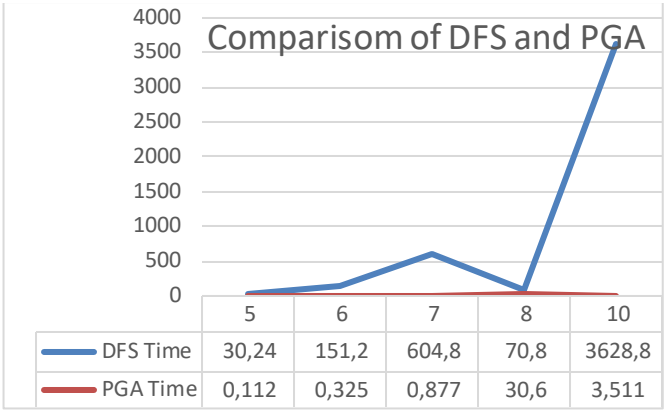

Figure 7. Comparison of DFS and PGA run times

8. CONCLUSION

This paper focused on the effective method of Cryptarithmetic problems. The use of the parallel genetic algorithm it has been shown that within an acceptable time, we can also find the result of large instances of this problem. The proposed algorithm uses multiple processors for scheduling with centralized control.Tasks are taken as batches and scheduled to minimize the time of execution and balance the processors' load. Clearly, by systematically searching for possible assignments of values to the letters by deducting search space, we can easily find solutions to cryptarithmetic problems.

Furthermore, the operation can be extended to subtract, multiply, division and we can have more than two operands.

ACKNOWLEDGEMENT

First and foremost, I'd like to thank my advisors, Daw Yi Yi Win, Daw Si SiSann and DawKhaingZarMyint Aung, for their guidance, motivation, enthusiasm, and profound experience during the research and writing this paper.

REFERENCES

[1] Vinod Goel, Sketches of thought, MIT Press,1 995, pp.87 and 88.
[2] Bonnie Adverbach and Orin Chein, Problem Solving Through Recreational Mathematics, Courier Dover Publications, 2000, pp. 156.
[3] S.MounirAlaoui, O. Frieder, and T.El-Ghazawi, "A Parallel Genetic Algorithm for task mapping on parallel machines". IPPS/SPDP Workshops 1999.
[4] R.Neduchelian, K.Koudhik, N.Meiyappan, V.Raghu, "Dynamic Task Scheduling Using Parallel Genetic Algorithms for Heterogeneous Distributed Computing", Proceeding of 2006 International Conference on Grid, Las Vegas, Nevada, USA,2006.
[5] Man Lin, Chen Ding, "Parallel Genetic Algorithms for DVS Scheduling of Distributed Embedded Systems", HPCC,2007.
[6] Abu Sayef Md. Ishaque, Md. Bahlul Haider, Muhammed Al Mahmud Wasid, Shah Mohammed Alaul, Md. Kamrul Hassan, Tanveer Ahsan, Md. Shamsul: "An Evolutionary Algorithm to Solve Cryptarithmetic Problem". International Conference on Computational Intelligence 2004: 494-496.
[7] MM Naoghare, VM Deshmukh: "Comparison of Parallel Genetic Algorithm with Depth First Search Algorithm for solving Verbal Arithmetic Problems". International Conference and Workshop on Emerging Trends in Technology (ICWET 2011)-TCET.
[8] David Goldberg, "Genetic Algorithms in Search, Optimization and Machine," Addison-Wesley, Reading, MA 1989.
[9] Mariusz Nowostawski, Riccardo Poli, "Parallel Genetic Algorithm Taxonomy", KES'99.
[10] Konfrst, Z, "Parallel Genetic Algorithms: Advances, Computing Trends, Applications and Perspective," Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International 26-30 Apirl 2004.

BIOGRAPHIES (Optional not mandatory)

Zin Minn Thant
Lecturer
Faculty of Computer Science
University of Computer Studies, (Kalay), Myanmar

Si SiSann
Lecturer
Faculty of Information Technology Support and Maintenance
University of Computer Studies, (Kalay), Myanmar

Yi Yi Win
Lecturer
Faculty of Computer Science
University of Computer Studies, (Pakokku), Myanmar

KhaingZarMyint Aung
Lecturer
Computer Department
Myanmar Aerospace Engineering University