A HEURISTIC APPROACH FOR OPTIMIZING TRAVEL PLANNING USING GENETICS ALGORITHM

Md. Lutful Islam[1], Danish Pandhare[2], Arshad Makhthedar[3], Nadeem Shaikh[4]

[1]*Assistant Professor, Computer Engineering, MHSSCOE, Maharashtra, India*
[2] *Student, Computer Engineering, MHSSCOE, Maharashtra, India*
[3] *Student, Computer Engineering, MHSSCOE, Maharashtra, India*
[4] *Student, Computer Engineering, MHSSCOE, Maharashtra, India*

Abstract

In today's fast-paced society, everyone is caught up in the hustle and bustle of life which has resulted in ineffective Planning of their very important vacation tour. Either they spend much time on deciding what to do next, or will take many unnecessary, unfocused and inefficient steps. The main purpose of our project is to develop a Travel Planner that will allow the customer to plan the entire tour so that he visits many places in less time. The concept would be implemented using Genetics Algorithm of Artificial Intelligence which would be used as a search algorithm to find the nearest optimal travel path. Moreover, In order to reduce the running time of GA, Parallelization of Genetics Algorithm would be demonstrated using Hadoop Framework.

Key Words: *Genetics Algorithm, TSP, Hadoop, and MapReduce etc…*

----------***----------

1. INTRODUCTION

Travel Planning has become an emerging topic and has attracted much attention in recent years. It is becoming increasingly important to provide travellers with craving information to assist them in the tour. Travellers, whether leisure -tourists- or business travellers will want on top of the attractions that a given destination offers, a flexible and convenient transport means to reach it. So the purpose of this Travel Planning System is to provide the traveller with optimal, feasible and personalized route between origin and destination [1].

The Travel Planning System is quite allied to TSP. So we would be implementing this by ruminating the Travelling Salesman Problem. The TSP deals with finding a route covering all cities so that the total distance travelled is minimal. There are mainly three reasons why TSP has attracted the attention of many researcher's and remains an active research area. First, a large number of real-world problems can be modelled by TSP and it falls in distinguished category of hard problems. Second, it was proved to be NP-Complete problem and cannot be solved exactly in polynomial time. Third, NP-Complete problems are intractable in the sense that no one has found any really efficient way of solving them for large problem size. Also, NP-complete problems are known to be more or less equivalent to each other; if one knew how to solve one of them one could solve the lot [2].

Conventional Linear Programming methods do find Optimal Solution to TSP but are infeasible and impracticable as the time required to obtain the solution is very huge. Since real time applications of TSP has higher problem complexity as it consists of hundreds and thousands of nodes, the primary objective of obtaining exact optimal solutions have been shifted to obtaining heuristically good solutions.

Genetics Algorithm (GA) is one such Heuristics based Algorithm that tends to find good and novel solutions to TSP within reasonable time. GA's are relatively new paradigms in artificial intelligence which are based on the principles of natural selection [3]. Based on a predefined Fitness value, GA iteratively goes on detecting the fittest solution which increases its performance thereby making it more suitable for finding solutions for many optimization problems.

One of the key points of GA is its ability of being parallelized. Thus, the Travel Planner will attempt to minimize the runtime of Travelling Salesman Problem using Genetics Algorithm by parallelizing it on Hadoop. Hadoop's MapReduce Framework provides better scalability, robustness, fault tolerance and easy means of access and use.

2. BACKGROUND

Before delivering the implementation details of how to integrate MapReduce and Genetic algorithms, discussion regarding certain concepts is highly significant. The following shows a brief overview of various topics.

2.1 Travelling Salesman Problem

Many mathematicians and Computer Scientists have immensely addressed the Travelling Salesman Problem which is considered to be the basic, fundamental and one of the toughest problems in Computer Science and Operations Research. The first instance of the traveling salesman problem was from Euler in 1759 whose problem was to

move a knight to every position on a chess board exactly once. The traveling salesman first gained fame in a book written by German salesman BF Voigt in 1832 on how to be a successful traveling salesman [4]. Its importance stems from the fact there is a plethora of fields in which it finds potential applications such as automatic drilling of printed circuit boards, threading of scan cells in a testable VLSI circuit, X-ray crystallography, DNA fragments and many more.

It can be normally defined as :-
1. Known Facts: A network of 'n' cities named {c1, c2, c3..., cn} with c1 as the starting node or city. A Cost Matrix 'C'= [Cij] denoting the calculated Euclidean distances between the cities 'i' and 'j'.
2. Constraints: Each city should be visited exactly once.
The traveller should return to the starting city.
3. Problem: To find the least cost Hamiltonian cycle. Thus, the aim is to obtain an optimal path cost such that the final sum of all distances between each node and its successor is minimized. An important point to note is that the first node becomes the successor of the last node in order to satisfy the constraints of the problem.

$$d = \sqrt{(|x_1 - x_2)^2 + (|y_1 - y_2|)^2}$$

The distances in the cost matrix can be Straight Line Distance, Euclidean distance or City Block (Manhattan) distance. Given two cities with co-ordinates c1=(x1, y1) and c2=(x2, y2), the Euclidean distance say 'd' can be calculated as:

The Travelling Salesman Problem has some variations which can be observed from the cost matrix 'C'.TSP is said to be Symmetric if Cij equals Cji for all values of 'i' and 'j'. It will be asymmetric otherwise. Thus, if a TSP problem with n-cities is considered then there will (n-1)! Possible solutions. Out of which the one with the minimum cost has to be selected. This becomes extremely difficult and impracticable as the number of possible solutions becomes too large even for a moderate value of 'n'.

2.2 Genetics Algorithm

Genetic Algorithms (GAs) are adaptive heuristic search algorithm premised on the evolutionary ideas of natural selection and genetics. As such they represent an intelligent exploitation of a random search used to solve optimization problems. Although randomized, GAs are by no means random, instead they exploit historical information to direct the search into the region of better performance within the search space. The basic techniques of the GAs are designed to simulate processes in natural systems necessary for evolution, specially those follow the principles first laid down by Charles Darwin of "survival of the fittest" [5].

Genetic algorithms are based on the principle of Genetics and Evolution which implies that the fitter individuals are more likely to survive and have a greater chance of passing their good genetic features to the next generation [6]. "Genetics" is derived from Greek word "genesis" meaning „to grow" or „to become". GA was first pioneered by John Holland in 1975 in the book "Adaptation in Natural and Artificial System" [7].

2.2.1 Why Genetics Algorithm?

GA is used since it is better than conventional AI systems. Unlike older AI systems, they do not break easily even if the input changes slightly, or in the presence of reasonable noise. Also, in searching 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, heuristics, depth-first, breadth-first, and praxis) [5]. Not only does GAs provide an alternative methods to solving problem, it consistently outperforms other traditional methods in most of the problems link. Many of the real world problems involved finding optimal parameters, which might prove difficult for traditional methods but ideal for GAs. The appeal of GAs comes from their simplicity and elegance as robust search algorithms as well as from their power to discover good solutions rapidly for difficult high-dimensional problems. GAs are useful and efficient when the search space is large, complex, or poorly understood, domain knowledge is scarce or expert knowledge is difficult to encode to narrow the search space, no mathematics analysis is available and when the traditional search methods fail [8]. The continuing improvement in the performance value of GA's has made them attractive for many types of problem solving optimization methods. In particular, genetic algorithms work very well on mixed (continuous and discrete) combinatorial problems. Moreover they are also less susceptible to getting 'stuck' at local optima than gradient search methods[3].

2.2.2 Working

GA begins by generating an initial population which is called as 'genome' or a 'gene pool' which will then be used to generate successive generations. This is achieved by application of three genetic operations to generate new and better population as compared to the older generations. A fitness score is assigned to each solution representing the abilities of an individual to 'compete'.

The very first operator is Selection operator in which the individual with the optimal (or generally near optimal) fitness score is sought. Parents are selected to mate, on the basis of their fitness, producing offspring via a reproductive plan. Consequently highly fit solutions are given more opportunities to reproduce , so that the offspring inherit characteristics from each parent [5].

The second operator is Crossover wherein intermixing of alleles of two parents are performed to obtain the new offspring.

The third operator is Mutation that maintains uniqueness and diversity of the gene pool in order to prevent any loss of

genetic information caused due to crossover. The process repeats until an optimal solution is discovered. Individuals in the population die and are replaced by the new solutions ,eventually creating a generation once all mating opportunities in the old population have been exhausted. In this the better solutions thrive while the least fit solutions die [5].

3. PROPOSED GA FOR TSP

The aim of this paper is to develop a software for proper panning of the entire tour. This will implicitly include solution of Travelling Salesman Problem using Genetics Algorithm. Details of different components and operators required are as follows:

3.1 Initialization of the Gene Pool (Population)

A random population is initially generated that consists of some tours associated with their respective costs. The individuals in the pool should be different from each other in order to maintain the population diversity. Simultaneously, there should also be a control on the population size in order to improve the performance of the Algorithm. Therefore one constructs tours with a greedy heuristic and improve this by a tour improvement heuristic [9].

1. Fitness Function: There are two different way of calculating fitness value of a particular chromosome.

GAs are used in maximation problem but as TSP is a minimization problem i.e. minimium cost path is chosen reciprocal of the maxima function can be used for deciding the fitness value of an organism. Thus if 'fi' denotes the fitness values and 'Di' represents the maximum total Distance then fitness function can be stated as

$$fi = 1/ Di \ldots(i)$$

A fitness function evaluation is incorporated to assigns a value to each organism, noted as fi. This fi value is a figure of merit which is calculated by using any domain knowledge that applies. In principle, this is the only point in the algorithm that domain knowledge is necessary. Organisms are chosen using the fitness value as a guide, where those with higher fitness values are chosen more often. Selecting organisms based on fitness value is a major factor in the strength of GAs as search algorithms. The method employed here was to calculate the total Euclidean distance Di for each organism first, then compute fi by using the following equation

$$fi = D\,max - Di \ldots(ii)$$

where Dmax is the longest Euclidean distance over organisms in the population [3].

3.2 Selection and Survival of the fittest

The selection operator chooses two members of the present generation to participate in the next operations of crossover and mutation.[3] Two different approaches for selection are

1. Roulette Selection: Assign a probability to each organism i, computed as the proportion using the following equation

$$Pi=fi \div \Sigma fj \quad \ldots.\{iii\}$$

where j=1, 2, 3... n! [3].

2. Deterministic Sampling: Assign to each organism 'i', a value Si evaluated by the relation.

$$Si = ROUND\,(Pi * POPSIZE) + 1 \quad \ldots.(iv)$$

(Where: ROUND means rounding off to integer and POPSIZE means the population size).The selection operator then assures that each organism participates as parent exactly Si times [3].

Table -1: Fitness values and Selection Criteria

	Individual Chromosome	Di	fi (as per eq.	fi (as per eq. (ii)	Pi	Si
1	BCDEFA	255	0.003	120	0.133	2
2	DFEABC	175	0.005	200	0.222	2
3	CBDAEF	95	0.010	280	0.311	3
4	FEDCBA	375	0.002	0	0.001	1
5	ABCDEF	75	0.013	300	0.333	3
				$\Sigma f_i = 900$		

It can easily be inferred that no matter which rule is used 5th and 3rd individuals have higher fitness value as compared to others and hence these two organisms will become parent and further process continues as follows.

3.3 Crossover

Crossover is a process of generating an offspring from selected two parents. The genes of the both the parents are converged and combined in order to obtain two children. Following are the different types of crossover operators.

3.3.1 Single Point Crossover

Only one cross point is used. Children are generated by swapping the alleles after second cross point of both the parents.

Example :-

Parent 1: ABCG|HBDE	Child 1: ABCG|GCBH
Parent 2: ADEF|GCBH	Child 1: ADEF|HBDE

3.3.2 Two Point Crossover

Two cross over points are used. Swapping any the middle section will result in the production of new child. From the examples it can be clearly seen that both Single point and two point crossovers may produce an invalid child. Some cities are visited more than once while some are not even visited at least once.

Thus, some modifications are desirable in these crossovers in order to have valid organisms after reproduction.

Example:-

Parent 1: ABC|GHB|DE	Child 1: ABC|FGC|DE
Parent 2: ADE|FGC|BH	Child 1: ADE|GHB|BH

3.3.3 Order Crossover

To apply order crossover, two random cross points are selected. Alleles from parent1 that fall between the two cross points are copied into the same positions of the offspring. The remaining allele order is determined by parent2. Non duplicative alleles are copied from parent2 to the offspring beginning at the position following the second cross point. Both the parent2 and the offspring are traversed circularly from that point.

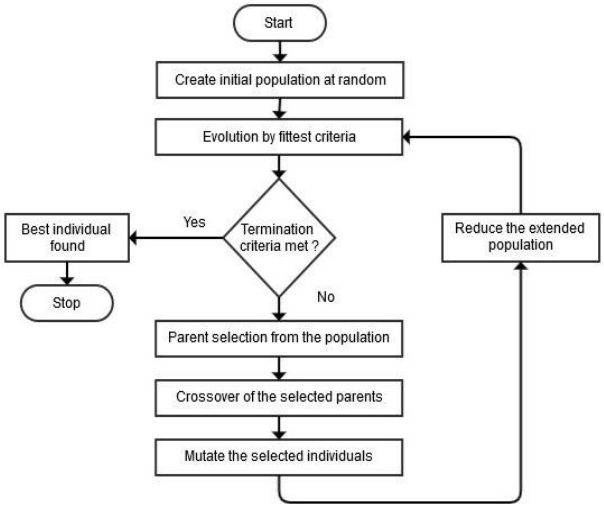

Fig -1: Genetic Algorithm Flowchart

A copy of the parent′s next non duplicative allele is placed in the next available child position. An Example of OX is given below with two random cross points; 3 and 6, the alleles in the crossing sites from parent1 (GHB) are copied into the same positions of the child. The alleles after second cross point in parent2 (BA), B is skipped since it already exists in child; therefore only A is copied to child at position 7. Traverse parent2 circularly, the alleles (HDE), skip H to D which is copied to child at position 8. Traversed Child circularly, the alleles from parent2 (EF) are copied to child at positions 1 and 2. Finally, skip G to C and copy it to child at position 3 [9].

Example :-

Parent 1: ABC|GHB|DE	Child 1: EFC|GHB|AD
Parent 2: HDE|FGC|BA	Child 1: ABH|FGC|DE

3.3.4 Partially Matched Crossover

PMX proceeds just as OX. Alleles from parent1 that fall between two randomly selected crossing sites are copied into the same positions of the offspring. The remaining allele's positions are determined by parent2 during a two step process. First, alleles in parent2 not within crossing sites are copied to the corresponding positions within the offspring. Next each allele of parent2 within the crossing sites is placed in the offspring at the position occupied in parent2 by the allele from parent1 that displaced it. See the example below. The random crossing points are 3 and 6, the alleles in the crossing site of parent1 (AHG) displace in the child (DEF), and then the alleles (BC) in parent2 which are not in crossing site are copied into the same positions of the child. D goes to position 1 which is the position with respect to parent2, displacing allele A. Finally E and F are placed in the positions of H and G, respectively [9].

Example :-

Parent 1: FBC|AHG|DE	Child 1: DBC|AHG|FE
Parent 2: ABC|DEF|GH	

3.4 Mutation

In order to maintain the uniqueness and diversity in the generated population. Mutation just alters the result by a small amount. It randomly selects a location and performs some modification. This modification can be swapping of a single bit or inverting a bit value.

3.4.1 Need for Mutation

Organisms with less fitness values are discarded after every successful iteration. This elimination process might result in some loss of genetic information. Random occasional modification by means of mutation ensures maintenance of that lost genetic information which selection and crossover cannot guarantee. Thus, Mutation helps in preventing the loss of genetic information, maintains uniqueness and ensures diversity in the newly generated population.

4. PARALLEL GA BASED ON HADOOP MAPREDUCE

In this section we present the design of the proposed parallel Genetics Algorithm using Hadoop Map Reduce.

4.1 Parallelizing Genetics Algorithm

Genetics Algorithm as stated earlier is an effective mechanism to find acceptable solution to problems in business, engineering and science. GA's are generally able to find satisfactory solutions in reasonable amount of time, but as they are applied to more complicated and bigger problems there is an increase in the time required to find an

acceptable solution. However their exist some problems in their utilisation part which are as follows :-

1. For some kind of problems, the population needs to be very large and the memory required to store each individual may be considerable (for e.g:- genetics programming). In such cases running and application using a single GA is inefficient, so parallel form of GA is necessary.

2. Sequential GAs may get trapped in a sub-optimal region of the search space thus becoming unable to find better quality solutions. PGAs can search in parallel different subspaces of the search space , thus making it less likely to be trapped by low quality sub-spaces [11].

As a consequence, multiple efforts are made to make GA faster, and one of the most promising choice is to use parallel implementation of GA. So the parallelization of the Genetics Algorithm is the most significant aspect of the project and the parallelization of the genetic algorithm would ultimately reduce the run time of the algorithm. GA falls into three different classes which includes Global, Coarse-Grained, Fine-Grained and Hybrid. Global PGAs use a single population and simply parallelize the evaluation of the fitness, and then sequentially produce the next generation. Coarse-grained parallelization involves evolving many separate subpopulations, called demes, in parallel. Coarse-grained PGAs often also implement migration, allowing an individual to move from one deme to another. Fine-grained parallelization involves assigning one individual per processor core. This is usually undertaken with special hardware. Hybrid PGA is the combination of all the above techniques [12]. Also, the advantages of using various PGAs facilitates no changes in the traditional GA. Parallel Genetic Algorithms divide the search space into many smaller pieces to find the near optimal solution. During this process the sub-optimal solutions need to avoid local minima. Different frameworks are available for parallelization such as Hadoop, Open-cl, Parallel Java, and C-MPI. Among all these frameworks this project would be implemented using Hadoop which is the most popular and flexible one.

4.2 Hadoop

Hadoop is an Apache project being built and used by a global community of contributors, using the Java* programming language. It doesn't maintain indexes or relationships; you don't need to decide how you want to analyze your data in advance. It breaks data into manageable chunks, replicates them, and distributes multiple copies across all the nodes in a cluster so you can process your data quickly and reliably later. Hadoop is also use to conduct analysis of data [12]. It is licensed under the Apache 2.0. Hadoop is very scalable and has a robust file system which is designed to handle all the intra-communication between different nodes in a cluster. It is fault tolerant and recovers from failure if one of the data nodes suffers from failure. Hadoop framework consists of two main layers Hadoop Distributed File System (HDFS) and Execution engine (MapReduce). Hadoop introduce many components like MapReduce, Hive, H-base, HDFS, Pig, Chukwa, Avro, Hive, ZooKeeper etc [12].Yahoo!, has been the largest contributor to this project, and uses Apache Hadoop extensively across its businesses. Other contributors and users include Facebook, LinkedIn, eHarmony, and eBay.

4.2.1 MapReduce Framework

The MapReduce framework was developed by Google. It enables highly fault-tolerant massively scalable computation to occur on a network of commodity hardware. MapReduce has proven successful at allowing computation over terabytes of data to become routine [12]. MapReduce' is a framework for processing parallelizable problems across huge datasets using a large number of computers (nodes), collectively referred to as a cluster or grid. Computational processing can occur on data stored either in a files system (unstructured) or in a database (structured). MapReduce is an elegant and flexible paradigm which enables to develop large-scale distributed applications [13].

4.2.2 Outline of MapReduce

The MapReduce model works by splitting large tasks into smaller tasks. The smaller tasks are executed in parallel. The Map Reduce framework handles the Intra cluster communication. Map Reduce has two important steps, the map and reduce function. The data flows from the mapper to the Reducers. The parallelism occurs in the mapper phrase. The data is pre-processed; it's converted to key value pairs. The key value pairs are passed to the map function. The results from different mappers are combined or merged in the reducer phase. A reducer takes in a key value pair and produces a collection of new values [14]. All the fundamental steps of Genetics Algorithm are carried out in parallel and the above process is repeated till the optimal value is reached.

4.3.3 Logical View of MapReduce

The Map and Reduce functions of MapReduce are both defined with respect to data structured in (key, value) pairs. Map takes one pair of data with a type in one data domain, and returns a list of pairs in a different domain: Map(k1,v1) → list(k2,v2). The Map function is applied in parallel to every pair in the input dataset. This produces a list of pairs for each call. After that, the MapReduce framework collects all pairs with the same key from all lists and groups them together, creating one group for *Logical view of Map Reduce:* The Map and Reduce functions of MapReduce are both defined with respect to data structured in (key, value) pairs. Map takes one pair of data with a type in one data domain, and returns a list of pairs in a different domain: Map(k1,v1) → list(k2,v2). The Map function is applied in parallel to every pair in the input dataset. This produces a list of pairs for each call. After that, the MapReduce framework collects all pairs with the same key from all lists and groups them together, creating one group for each key [15].

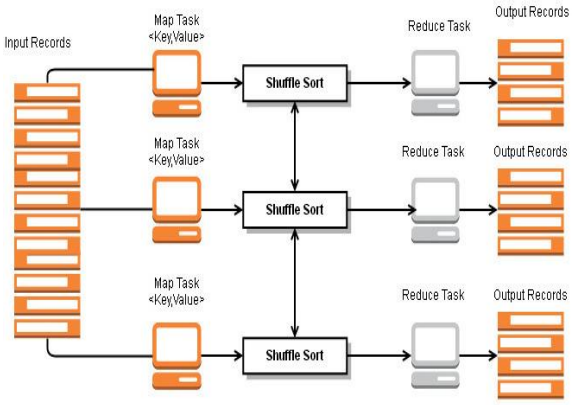

Fig -2: Hadoop MapReduce Model

The Reduce function is then applied in parallel to each group, which in turn produces a collection of values in the same domain: Reduce(k2, list (v2)) → list(v3). Each Reduce call typically produces either one value v3 or an empty return, though one call is allowed to return more than one value. The returns of all calls are collected as the desired result list. Thus the MapReduce framework transforms a list of (key, value) pairs into a list of values [15].

CONCLUSION

In this paper we have discussed the Travel Planning System using Genetics Algorithm to assist the user to obtain optimal tour to visit n cities. It has also proposed the use of advanced operators of Genetic Algorithms in order to enhance the rate of divergence, and achieved tours with reasonable time. The obtained results highlighted that using Parallel Genetics Algorithm allowed us to save more time. and the Map Reduce model helps to increase the processing speed. Certain aspects in the project such as the parallel design of GA can be enhanced in the future. An enhanced parallel design can incorporate more computation of selection, crossover or mutation in the mapper phase. This will put less pressure on the reducers. The architecture can be improved further by adding more reducers

REFERENCES

[1]. Konstantinos G.Zografos and Michael A.Madas, " A Travel and Tourism System Providing Readl-Time, Value Added Logistical Services on the Move", Athens University of Economics & Business(AUEB).

[2]. Dino Keco and Abdulhamit Subasi, " Parallelization of genetics algorithm using Hadoop Map/Reduce".

[3]. Buthainah Fahran Al-Dulaimi, and Hamza A. Ali, "Enhanced Travelling Salesman Problem Solving by Genetic Algorithm Technique (TSPGA) in paper 2008.

[4]. Kylie Bryant and Arthur Benjamin, "Genetics Algorithm and the Travelling Salesman Problem", thesis, Department of Mathematics, Harvey Mudd College, Dec. 2000.

[5]. Introduction to Genetics Algorithm (1996). [Online]. Available: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/hmw/article1.html.

[6]. Noraini Mohd Razali and John Geraghty, "Genetic Algorithm Performance with Different Selection Strategies in Solving TSP flwcchart," in *Proceedings of the World Congress on Engineering* 2011 Vol II WCE 2011,paper 06.08.11, London, U.K.

[7]. Ashish Gupta and Shipra Khurana, " Study of Travelling Salesman Problem using Genetics Algorithm",vol 2,issue 5,pp 575-588, May 2012.

[8]. Genetics Algorithm (1996) Web page on www.doc.ic.ac.uk. [Online]. Available: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol4/tcw2/report.html.

[9]. Konstantinos G.Zografos and Michael A.Madas, " A Travel and Tourism System Providing Readl-Time, Value Added Logistical Services on the Move", Athens University of Economics & Business(AUEB)..

[10].Varshika dwivedi, Taruna Chauhan, Sanu Saxena and Princie Agrawal, "Travelling Salesman Problem using Genetics Algorithm".

[11].Mariusz Nowastawski and Riccardo Poli, " Parallel Genetic Algorithm Taxonomy", *Proc*' 99,paper May 13,1999.

[12].Rahate Kanchan Sharadchandra and L.M.R.J Lobo, " Parallelization of Genetics Algorithm using Hadoop",vol 1,paper Nov-12.

[13].Linda Di Geronimo, Filomena Ferrucci, Alfonso Murolo and Federica Sarro, " A Parallel Genetics Algorithm Based on Hadoop MapReduce for the Automatic Generation of JUnit Test Suites".

[14].Akshat Mishra ,'Genetic Algorithm for the Travelling Salesman Problem on Hadoop",Tech. Rep. 2011.

[15].Map Reduce (2014) Webpage on Wikipedia [Online]. Available: http://en.wikipedia.org/wiki/MapReduce.

[16].Introduction to Genetics Algorithm (2014) Webpage on obitko [Online]. Available: http://www.obitko.com/tutorials/genetic-algorithms/ga-basic-description.php

BIOGRAPHIES

Md. Lutful Islam is an Assistant Professor at M.H Saboo Siddik College of Engg. His qualification includes M.Tech from Aligarh Muslim University and M.C.A from Rajasthan Vidyapeeth, Udaipur. He has a teaching experience of 15 yrs. His key areas of interests are Computer Graphics, Discrete Mathematics, Robotics and Artificial Intelligence.

Danish Pandhare is a final year student(B.E) of M.H.Saboo Siddik college of Engineering, Byculla, Mumbai University. He holds a Diploma in Computer Engineering from Vidya Prasarak Mandals Polytechnic, MSBTE. His key areas of interests include Web Designing, Cloud Computing , Networkng, Analysis of Algorithm and Design

Arshad Makhthedar is a final year student(B.E) of M.H.Saboo Siddik college of Engineering, Byculla, Mumbai University. He holds a Diploma in Computer Engineering from M.H Saboo Siddik Polytechnic, MSBTE. His key areas of interests include AI and Soft Computing, Mathematics and Statistics.

Nadeem Shaikh is a final year student(B.E) of M.H.Saboo Siddik college of Engineering, Byculla, Mumbai University. His key areas of interests include Distributed and Internet Computing Systems, Web Security, Web based Systems, and Web Security.