International Journal of Scientific Research in Computer Science, Engineering and Information Technology
ISSN : 2456-3307 (www.ijsrcseit.com)
doi : https://doi.org/10.32628/IJSRCSEIT

Implementation of Genetic Algorithm Using the Traveling Salesman Problem in Cloud

Aadil Bashir*[1], Manoj Kumar Srivastava[2]
*[1] M.Tech (Scholar), CSE Depatment Desh Bhagat University, Mandi Gobindgarh, Punjab, India
[2] CSE Department, Desh Bhagat University, Mandi Gobindgarh, Punjab, India

Article Info
Volume 7, Issue 4
Page Number: 350-358

Publication Issue :
July-August-2021

Article History
Accepted : 20 July 2021
Published : 27 July 2021

ABSTRACT

The paper below is utilised to create a novel cross-operator (SCX) for an algorithm which creates premium solutions for problem of travelling salesmen (TSP). In cross-operative sequential and constructive operator method it creates a new offspring from a parent with increased constraints depending on its standards, which may be found in the composition of parents while maintaining the parent chromosomes' node order. The Internet connects the entire world. Artificial intelligence (AI) is in high demand, thanks to the large number of web users and the growing popularity of cloud computing research. Through natural selection and genetic development, genetic algorithms (GA) are applied as an AI optimisation technique in this study. There are numerous GA applications, such as web mining, load balancing, routing and planning, and online service selection. As a result, determining whether code has a significant impact on GA server speed and web-based language technology is a difficult undertaking. The Travel Salesman (TSP) specified as a Non Polynomial-Hard difficulty with the aim can be solved with GA (NP-hard). Although many academics use Python to implement GA, other popular high-level programming languages for interpreters, such as PHP, are also often used (PHP Hypertext Preprocessor). Different programming languages had different line of GA implementation and runtime codes, file sizes, and performance. The use of Python in GA implementation is suggested based on the findings.

Keywords : NP-complete, problem of a travelling salesman, an algorithm based on genetics, Cross-web-program sequencing and constructive.

I. INTRODUCTION

The (TSP) problem is a standard and an aged problem in computer science and operational research (TSP). It's possible to put it this way: A network with 'n' nodes, and trip costs (journey instance, or distance, etc.) is provided, as well as a matrix A= [Aij] of order n related with node i j). This network will be made available. The goal is to find the Hamiltonian cycle with the least amount of money. Based on the structure of the cost matrix, TSPs are: Asymmetries and Symmetry. When Aij = Aji, for i j, TSP is symmetrical; otherwise, it is asymmetric. In case of n-city Asymmetric , there are approx (n-1)! Solutions, in which one or other are at lowest rate. In any case, the number of answers for even a large n accumulates to the point where a thorough search is impossible. TSP has piqued the interest of many scholars and is a hot topic of study for three main reasons. To begin with, TSP may simulate a wide range of real-world challenges. Second, the existence of the NP-Complete problem has been demonstrated [1]. The, problem of NP-complete are persistent in the intellect that no efficient solution identified for enormous problem dimensions. Furthermore, it is well known that NP-complete problems are more or less equal; if you can solve one, you can solve the others. TSP is also useful in a variety of applications, such as automatic boiler boards and scan cell threading on a tested circuit[2], X-ray [3], and so on. Methods which endow with a solution for problem are known as accurate actions. Simply listing and evaluating the objective function values of all potential solutions and selecting the best is an implied strategy for resolving the TSP. However, due to the vast number of viable TSP solutions, even for moderate-scale problems, this "exhaustive search" is clearly wasteful and impracticable. Because real-world applications necessitate the resolution of substantial challenges, the focus has shifted to finding 'good answers' in a heuristic fashion within an acceptable timeframe, 'the degree of goodness,' with the goal of finding perfectly optimum solutions for TSP. Among the best heuristic strategies for resolving TSP instances is the genetic algorithm (GA). Because the crossover operator is so important in GA, the TSP has had multiple crossover operators. Because data processing and analysis scripts are often time consuming and need many hours of computation on a computer device, the iteration and debugging processes will be longer [5]. In addition, scientists emphasise their work differently than professional programmers. They are more concerned with the process than with the instruments they employ. Many scientists and even inexperienced programmers who consider themselves to be competent aspire to complete programming chores more quickly. They'll use that programming language, of course (PL). It is also based on a psychological assessment, as detailed in [29]. AI programming must be quick and simple in order to stay up with the quickly growing results in AI. As a result, scientists need PLs that can iterate quickly while keeping order and clarity so that they can be used easily.

Even if compiler PL is faster than interpreter PL while running programmes, Python[25] and PHP[26] are less easier than newer PLs. The virtues of Python are mainly its ease of use, its interpretations or its object-oriented programming language which, in keeping an object-oriented manner, may meet many scientific needs. On the other hand, the effectiveness of PL can be assessed by the number of lines of code or syntaxes needed to achieve the same AG. In denial of service assaults [30] another problem in using built-in PL is evident.

II. Literature Survey

Two crossover points were used by the operator dubbed PMX, Goldberg and Lings [4] defined the operator. Interchange mapping is defined by the section that connects these places. A well-known 33-node problem was almost optimised in this first attempt at applying GAs to the TSP.

It selects the tour sequence of one parent and keeps the node n order of the other parent. Davis[5].

Oliver et al. [6] presented another crossover operator called CX (cycle crossover) operator, which generated offspring so that each node (and position) derives from a parent.

Whitley et al. [7] suggested a crossover edge recombination operator (ERX) using a 'edge map' for building an offspring in which information from parents' structures can be obtained as much as feasible. This edge map saves all the connections that lead in and out of a node from both parents.

Radcliffe and Surry presented a N-point crossover operator (GNX).[8]

PMX operator added to TBX by Choi et al.[10].

Moon et al [11] proposes the novel Moon Crossover operator (MX), which imitates moon variations, such as moon-waxing half moon — all of them full moon — to be known as gibbous. The performance of MX and OX operators is approximately equal, however for all the experiments OX never achieved an optimum solution. A new crossover operator dubbed SCX is introduced here and therefore, It is possible to solve the TSP using a genetic algorithm based on SCX.

A huge number of tenants share resources in the cloud. Fairness of resources is then studied by multiple users [13, 14]. There is a multi-resource distribution technique (known as DRFH)[13] which ensures that cloud-users with heuristics use resources fairly.

In huge datacenters with tens of thousands of servers, resource efficiency is becoming highly important[15, 16]. Some approaches, such as memory[17] and I/O[18], are intended to improve the use of computational resources. Application SLAs have been introduced in a number of ways [19]. Special applications include streaming [20, 21] and the business process [22]. Some resource management solutions are also presented.

Cloud resources are generally rented under a model that you can pay for. Many scholars have investigated the economic efficiency of cloud computing[23, 24]. The demand response trading systems are meant to reach the highest possible level of social welfare randomly. In this paper, our work focuses mostly on VM positioning stability. Due to the time-varied workloads in mobile cloud computing in particular, stability increases.

Problem Formulation

Many investigations are using GA for benchmarking applications[27]. This document sets benchmarks for PL in AI support. GA's problem area is the Travel-ling Salesman's Problem, in which numerous heuristics have formed a benchmark for GA performance testing[28]. The utilisation of TSP depends on the problem in its domain. TSP is an NP-hard issue that is optimised by GA to address NP's entire domain problem based on natural selects and genetic developments.

At TSP, it is the aim to identify a route for a certain number of towns by visiting every town exactly once and to return to the beginning town where the route is kept to a minimum. One way that goes back to every city to make a closed route is called a route. It goes back to the early city. The easiest and most straightforward way to solve TSPs is to list each route, to calculate the length of each routing and to select the shortest route.

Genetic Algorithms

Genetic algorithms (GA) are mainly based on a random alteration in the chromosomal genestructures of evolutionary biology to replicate the survival of the fittest among the species[12]. Two major needs must be fulfilled to resolve any real life problem by GA: An objective fitness function and hence a fitness function can be used to determine whether a solution is excellent or not. A simple GA operates by producing a new, and presumably better population than successive generations through random creation of an initial string population, known as the gene pool then using (three) operators. One can use the first operator to recreate strings with an appropriate degree of probability in subsequent generations, using their objective function values. As a result, new strings are formed. The second operator is the crossover, where randomly selected string pairs are joined. 3rd operators are used to change the value of a string on occasion. As far as the GA search process goes, crossover and reproduction is the most powerful. In addition, mutation broadens the search area and minimises the loss of genetic material for reproduction and crossbreeding. The possibility of exploiting a mutation is consequently quite little, whereas the likelihood of crossover is elevated.

Genetic coding

Encoding answers as viable chromosomes is necessary to make feasible chromosomes viable.. The technology of encoding solutions varies depending on the situation and involves some art. For the TSP, the chromosome-length solution is often the number of nodes. Each chromosome gene receives a node label such that there is no node in the same chromosome twice. Both adjacency representation and track representation are commonly used to describe the TSP tour. As an example, let's look at a tour that displays only the node label as a path. Let 1, 2, 3, 4, 5 be labels of nodes and can be shown as a tour (1, 3, 4, 2, 5).

Reproduction operator

As part of the selection procedure, chromosomes are duplicated into the next-generation mattress with a probability associated with their fitness value. A larger share of the best suited chromosomes is allotted the following generation in Darwinian reproduction. Fitness is determined in nature through the capacity of an organism to survive predators, pesticides and other adult reproductive barriers. No new chromosomes are generated in this period. The generally utilised playback function is the proportionate playback operator, when a string is chosen with a probability proportional to the fitness of the matting pond.

The SCX

As new chromosomes are produced from old ones, the solution space is sought. Crossover is the single most important search in the world. First, a couple of parents are selected randomly from the mattress. Second, a location known as the crossover site is randomly selected along the common length of the site and information is exchanged after the crossover site of both parent strings and two new children are created. Naturally, the TSP does not support this fundamental crossover strategy. The SCX produces a heir utilising superior borders based on the values in the parents' structure. The superior vertices that are not present in the structure of the parents are also utilised in the design process. Similar to the structure of parents in the case of Erx and Gnx, SCx creates new kindergarten edges that do not exist in the current population. It is therefore more likely that a better offspring than those of ERX and GNX will be born. In an early version, the operator is cited as the local strategy for enhancing [14, 15]. The SCX algorithm consists of:

Step 1: - Begin at 'node one' (for example, p =1)..

Step 2: Sequence the two parent chromosomes to look for the first 'valid node' Sequence

Appeared in each parent following 'node p' (the node not yet visited). Sequentially seek { 2,3, ...,n{regard as the first "valid" nodule in a single parent when p, node is found, to set off through step 3. Suppose that in the 1st and 2nd parents the 'Node α' and the 'Node β' are identified, then move to step 4 to find a next node.

Step 4: If cpα < cpβ, select the 'Node α' as the next node and link it to the partially formed daughter chromosome. If the offspring is a whole chromosome, otherwise stop renaming the current node as 'node p' and go to step 2.

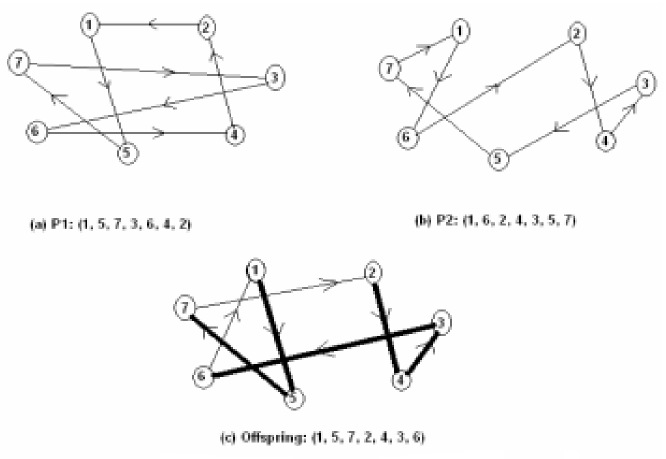
(a) P1: (1, 5, 7, 3, 6, 4, 2)
(b) P2: (1, 6, 2, 4, 3, 5, 7)
(c) Offspring: (1, 5, 7, 2, 4, 3, 6)

Figure 1: Example of SCCO

Survivor selection

After the crossover process, the survivor selection method is utilised to select the next generation population. Typically, only the fitter chromosomes are taken into account by the survivors of GA. Two types of GA chromosomes are taken into account in the survivor selection of the following generation: (1) parents in the current size m population and (2) descent created by size m cross-over. The selection technique (μ+α) of survivors combines chromosomes into (1) and (2), arranges chromosomes in upwards according to fitness and consider the future generation's first m chromosomes. In the worst situation, all μ parents will survive in the following generation.

Mutation operator

The mutation operator selects a chromosomal location randomly and alters the
Related alone, changing information thereby. Mutation is needed because the inadequate members of subsequent generations are removed, so that certain aspects of the genetic material may never be lost. By carrying out infrequent random chromosome modifications, GAs ensure that new search area sections are accessed, which can not be fully guaranteed by reproduction and crossover alone. This ensures that no key characteristics are missed prematurely, hence preserving the diversity of the matrix pool. The TSP is not working with the conventional mutation operator. We evaluated the mutual exchange mutation in this survey, which randomly chooses two nodes and exchanges them.

Parameters of control

The GA search process is governed by these factors. Some of the following are: (a) Population size: - Determines how many genetic materials are accessible for the research and then how many chromosomes. When the search is too small, the space is not covered sufficiently. If too much is involved, the GA delays chromosomal evaluation.
(b) Probability for crossover: – Specifies the likelihood of crossover between two chromosomes. c) Probability for mutation: - The likelihood of making bit-wise mutation is specified. (d) Criteria for termination: - Specifies when the genetic search will end.

Structure of genetic algorithms

The following can be summarised: GAs: Random population initialization; population assessment; generation = 0; {generation = generation + 1 is not met when the termination requirement is not satisfied; Crossing with crossover probabilities (Pc); choose fitting chromosomes by selecting the survivor; perform mutation with mutation probability (Pm);; Population assessment; Something for TSP class implementation Initialize individual function (route) Start of a person consisting of cities or nodes Make function Chromosome Chrome (file) Generate a single file function chromosomal evaluate Using the Euclid formula, calculate the length of the population given (a number of cities) Crossover function (other) Recombine a new child from a specific step in a wife's function Make a jump Assess the fitness expenses of everyone in the current cross-over of population Candidate population initialization Choose from the present population two best parent candidates (Parent A and Parent B).. Something for TSP class implementation Initialize individual function (route) Initialization of a person consisting of cities or nodes While there is a still smaller pool of candidates, the crossover rate is random. when the rate of crossover is then descended = crossover parent A with parent B other descendants = copy parent A randomise mutation rate when the rate of mutation is then descended = current descended descendants assess descendants' fitness costs If there is no offspring in the candidate population then add (individual) offspring to the candidate population. Population candidates are changing the new population function (individual)

III. Methodology

In this research TSP-based GA is implemented in two extensively used cloud server-side languages without frames. It's Python, PHP. All GA codes representing each PL are written on the basis of the pseudo code given. The variable names, methods and logic of initialization were used for the implementation. The data were tested using one source of information and the identical values for all PL parameters during implementation.

Every PL's GA codes are as close as possible implemented. If a single PL uses numerous methods, functions or variables for the implementation of any section of the pseudo code, then the second PL should also be done in the same way. Maintaining the code closest to the measurement findings apparently results. A random number generator is one of the many important functions utilised in GA. In GA, random numbers are generating random numbers to compensate for the possibility that parents are cross-cutting, copying or mutating. A Pseudo-Random Number Generator function is used to generate the random number. PRNG is a random number generator feature that returns the same random value when it receives the same seed number. This methodology is used to ensure that all programmes are performed in the programme using the same method and loop, and that the same result is obtained. In this research, the PRNG function is implemented by means of a distinct script run by one system call every implemented PL.

A modest adjustment of the scripts was introduced in the performance measurement of the implemented PL with the addition of the current time or micromotion function at numerous points of code, while taking into account the best fitness cost. In order to confirm that all implemented scripts use the same seed number and random value, a test unit was used before any measures were performed using a predefined value and data, and all implemented PL would mate the same parents and produce the identical candidates in each generation. At the end of the script execution you should return the best person in the same generation.

Python, PHP built GA pseudo code for solving TSP in this research. All method and variables are unit tested to ensure that all implementations of the specified pseudo code have the same results and values. The codes use the same demographic data.

Due to the varying types of programming of each PL, methods and variables cannot be exactly implemented. All methods codes and variables applied throughout all PLs, however, are guaranteed to be consistent with the same values and are based on the same GA environment and population data.

IV. Result Analysis

Python, the PHP-codes result in the implementation of the GA pseudo code supplied. In 245, 300 lines of code, Python, PHP, PHP codes were implemented. The file size is 7981 bytes, accordingly Python, PHP, 6886 bytes. As Python scripters don't require closing tags on its method, function, or loop implementation like in PHP, the shortest line of codes is Python. But Python uses more bytes to implement when it comes to code file size.

Table 1

Number of Cities	PL	Maximum (ms)	Minimum (ms)	Average (ms)	Standard Deviation	Best Fitness Cost (Length)	Performance over PHP (%)
6	PHP	38,44	36,00	37,18	0,84	30,397	-5,53
6	Python	35,73	34,94	35,23	0,26	30,397	-
7	PHP	47,91	45,85	46,79	0,73	41,135	-1,56
7	Python	47,11	45,36	46,07	0,57	41,135	-
8	PHP	62,23	58,61	60,14	1,18	34,020	-6,09
8	Python	57,25	56,21	56,69	0,36	34,020	-
9	PHP	51,16	49,49	50,06	0,58	49,062	-3,56
9	Python	49,44	47,75	48,34	0,50	49,062	-
10	PHP	72,97	69,72	71,67	1,15	51,881	-2,63
10	Python	71,66	68,56	69,84	0,85	51,881	-

As the most frequently used python to conduct research, Python is utilised as the basis for performance measuring [11]. So we compare the running time of all PL with Python. In the field of web environment, PHP outperforms Python from research carried out by Jafar et al.[12].

On the basis of last seed, optimum fitness cost and the best measurement results from TABLE I, we can conclude that in the same generation all tests for the same number of cities return the best individual. This shows that all PL executions and their running flow are identical.

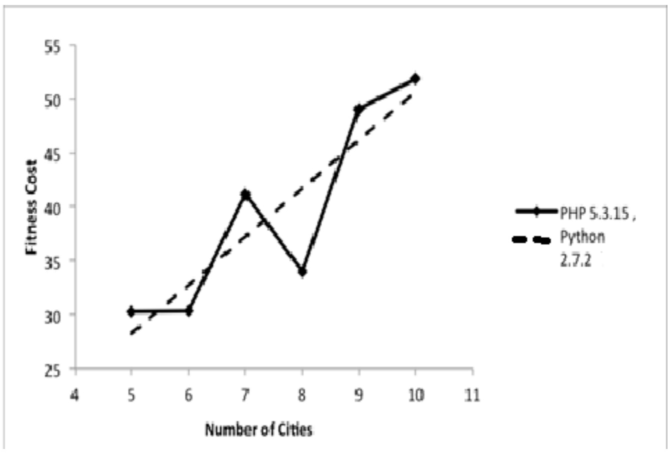

Fig. 2. Fitness cost as per number of cities

V. CONCLUSION

For a travellers' problem genetic algorithm We introduced a novel crossover operation called as the SCX crossover (TSP). Our main objective was to compare the effectiveness of remedies provided by various crossover operators. Our purpose was not to increase the optimal solution in any way. As a result, we don't use a local search technique to improve the method's quality. There is no large population size, and parallel versions of methods are not used to obtain a precise result, as Whitley et al. [7] did. To emphasise the true nature of crossover operators, the highest likelihood of crossover is also established. Lowest probability mutation is only used to avoid becoming stranded in local minima too quickly. It's tough to argue that this is exactly a moderated instance of our crossover operator. As a result, adding a capable localized search technique to the program can precisely resolve the other investigations. Data quantity (number of cities) in all PLs is measured and analysed in accordance with the programme execution time and the best potential fitness costs. The more the data, the longer it takes and the less time it takes for the GA result to become TSP

VI. REFERENCES

[1]. Fozia Hanif Khan, Nasiruddin Khan, Syed Inayatullah and Shaikh Tajuddin Nizami'SOLVING TSP PROBLEM BY USING GENETIC ALGORITHM'- International Journal of Basic & Applied Sciences IJBAS Vol: 9 No: 10.

[2]. IEEE paper by Mahdieh Poostchi on "A new Approach to Solve Traveling Salesman Problem Using Genetic Algorithm"

[3]. Crossover(Genetic Algorithm), http://www.wikipedia.org/genetic_algorithms.

[4]. Zakir H. Ahmed 'Genetic Algorithm for the Traveling Salesman Problem using Sequential Constructive Crossover Operator'- International Journal of Biometrics & Bioinformatics (IJBB) Volume (3): Issue (6).

[5]. Richard Johnsonbaugh , Marcus Schaefer, "Algorithms", Pearson Education, 2006 3rd edition.

[6]. Wikipedia: http://en.wikipedia.org/.

[7]. Genetic algorithms - A business perspective: Fritz H. Grupe and Simon Jooste (University of Nevada, Reno, Nevada, USA); Journal of Information Management & Computer Security.

[8]. http://lancet.mit.edu/.

[9]. http://www.iba.k.u-tokyo.ac.jp/ : Graduate School of Frontier Sciences, The University of Tokyo.

[10]. Practical Handbook of Genetic Algorithm Complex Coding System by Lance D Chambers.

[11]. H. Koepke. "10 reasons python rocks for research (and a few reasons it doesn't), " 2010.

[12]. Jafar, Anderson, and Abdullat. "Comparison of dynamic web content processing language performance under a LAMP architecture," West Texas A&M University Canyon, 2008.

[13]. W. Wang, B. Li, and B. Liang, "Dominant resource fairness in cloud computing systems with heterogeneous servers," in Proceedings of the IEEE Conference on Computer Communications (INFOCOM '14), pp. 583–591, IEEE, Toronto, Canada, May 2014.

[14]. J. Guo, F. Liu, J. C. S. Lui, and H. Jin, "Fair network bandwidth allocation in IaaS datacenters via a cooperative game approach," IEEE/ACM Transactions on Networking, vol. 24, no. 2, pp. 873–886, 2016.

[15]. D. Lo, L. Cheng, R. Govindaraju, P. Ranganathan, and C. Kozyrakis, "Improving resource efficiency at scale with heracles," ACM Transactions on Computer Systems, vol. 34, no. 2, 2016.

[16]. S. Singh and I. Chana, "QoS-aware autonomic resource management in cloud computing: a systematic review," ACM Computing Surveys, vol. 48, no. 3, article 42, 2016.

[17]. K. H. Park, W. Hwang, H. Seok et al., "MN-MATE: elastic resource management of manycores and a hybrid memory hierarchy for a cloud node," ACM Journal on Emerging Technologies in Computing Systems, vol. 12, no. 1, article 5, 2015.

[18]. R. C. Chiang, S. Rajasekaran, N. Zhang, and H. H. Huang, "Swiper: exploiting virtual machine vulnerability in third-party clouds with competition for I/O resources," IEEE Transactions on Parallel and Distributed Systems, vol. 26, no. 6, pp. 1732–1742, 2015.

[19]. N. Jain, I. Menache, J. Naor, and J. Yaniv, "Near-optimal scheduling mechanisms for deadline-sensitive jobs in large computing clusters," ACM Transactions on Parallel Computing, vol. 2, no. 1, 2015.

[20]. J. Ghaderi, S. Shakkottai, and R. Srikant, "Scheduling storms and streams in the cloud," ACM Transactions on Modeling and Performance Evaluation of Computing Systems, vol. 1, no. 4, 2016.

[21]. T. Wu, W. Dou, F. Wu, S. Tang, C. Hu, and J. Chen, “A deployment optimization scheme over multimedia big data for large-scale media streaming application,” ACM Transactions on Multimedia Computing, Communications, and Applications, vol.12, no. 5, article 73, 2016.

[22]. J. Xu, C. Liu, X. Zhao, S. Yongchareon, and Z. Ding, “Resource management for business process scheduling in the presence of availability constraints,” ACM Transactions on Management Information Systems, vol. 7, no. 3, article 9, 2016

[23]. L. Zhang, Z. Li, and C. Wu, “Dynamic resource provisioning in cloud computing: a randomized auction approach,” in Proceedings of the 33rd IEEE Conference on Computer Communications (’INFOCOM ’14), pp. 433–441, Ontario, Canada, May 2014.

[24]. Z. Zhou, F. Liu, Z. Li, and H. Jin, “When smart grid meets geodistributed cloud: an auction approach to datacenter demand response,” in Proceedings of the IEEE Conference on Computer Communications (INFOCOM ’15), pp. 2650–2658, IEEE, May 2015.

[25]. Python. http://www.python.org.

[26]. PHP. http://www.php.net

[27]. D. Dunlop, S. Varrette, and Pascal. "On the use of a genetic algorithm in high performance computer benchmark tuning," University of Luxem- bourg, 2008.

[28]. A. G. Najera. "TSP: three evolutionary approaches vs. local search," University of Birmingham, 2009

[29]. P. J. Guo and D. Engler. "Toward practical incremental recomputation for scientists: an implementation for python language, " Stanford Uni- versity, 2010.

[30]. S. Branigan. "Risk with web programming technologies, " Lucent Tech- nologies, 2000.

Cite this article as :

Aadil Bashir, Manoj Kumar Srivastava, "Implementation of Genetic Algorithm Using the Traveling Salesman Problem in Cloud ", International Journal of Scientific Research in Computer Science, Engineering and Information Technology (IJSRCSEIT), ISSN : 2456-3307, Volume 7, Issue 4, pp.350-358, July-August-2021.