International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075,Volume-8, Issue-9S, July 2019

An Scrutiny of Run-time Ramification for 5-Proviso Busy Beaver Proving Empirical Composition

Pardeep Singh Tiwana, Harjot Singh Tiwana, Rajeev Sharma, Astha Gupta

Abstract*:*The major aim of this paper is to undertake an experimental investigation for analyze the fluctuation between the descriptional (program-size) and computational time complexity. The investigation proceeds by systematic and exhaustive study for analysis of run-time complexity for 5-state Busy Beaver function using an experimental setup. To carry out experiment, TM simulator for Busy Beaver function will be tested for different N-values on different machines with different configurations and different platforms to calculate the run-time complexity. This study revealed that whether the Busy Beaver function is machine dependent. It also report that the average run-time of Busy Beaver function surely increases as the number of states.

Index Terms*: *Busy Beaver function, Computational complexity, Program-size complexity.

I. INTRODUCTION

There are plethora of methods to measure the computational complexity but some of them are concentrating on quantifying the resources like time, space and energy used by computation. The main objective of the research is to identify the relationship between the complexity measures, especially computational complexity and descriptional complexity. It has been thoroughly explained under the below subsection.

A. Computational complexity

Computational complexity is a branch of the theory of computations. It is study of understanding the difficulty of computational problems in terms of computational resources. It is used to measure how complex a problem is solved. Computational complexity of the problem is how many steps it takes to solve the problem using the most effective algorithm. It is the amount of time taken by the algorithm to run as a function of the length of the input representation. Basically, it is the number of resources (time or space) that it requires to solve a problem. Time complexity indicates the time taken by the solution of a problem.

B. Descriptional complexity

Descriptional complexity is also known as algorithmic or program-size complexity. Program-size complexity of a binary string is termed as the smallest program that can generate the string. There is no precise way which generates the shortest algorithm that produces a given string. The length of the shortest program of string is generally the complexity of a bit string. A string is said to be complex if the length of the string itself is much longer than its shortest description. If the shortest description can be much shorter than the length of the string itself, it is known as simple string.

C. Turing machines

Turing machine was concocted by Alan Turing in 1936. Turing machine was motivation of the PC framework that came two decades later. Turing machine was made for general calculation. It implies that Turing machine can register anything, which is processable [1]. Turing machine has two way interminable tapes which is separated into number of cells. Cell can either be clear or contain a non-clear image. Every cell contains just a single image. Turing machine has one head, known as R\W head (Read and Write head) that move over the phones of tape. R/W head can look at the one cell at once. At each progression, the machine peruses the image under the head, and relying on the present state, it compose new image in the cell under the head and goes to new state. The R/W head can either move left or right [1] [7].The primary advances pursued by a Turing machine:

A new symbol is written on the cell under the R/W head
A movement of R/W head: either head moves one cell left (L) or one cell right (R)
Goes to the new state
Whether to halt or not

Definition [1]: A Turing machine M has 7-tuple namely (Q, ∑, ┌, ∂, q0, b, F) where

1. Q is a finite non-empty set of states.
2. ┌ is a finite non empty set of tape symbols.
3. b ∈ ┌ is the blank.
4. ∑ is a non-empty set of input symbols and is a subset of ┌
5. ∂ is the transition function mapping (q, x) onto (q', y, D) where D is direction of movement of R/W head.
6. q0 ∈ Q is the initial state, and
F ⊆ Q is the set of final states [1] [7].

Revised Manuscript Received on June 15, 2019.
Pardeep Singh , IT, Chandigarh Engineering College, Landran, India.
Harjot Singh, CSE, Chandigarh Engineering College, Landran, India.
Rajeev Sharma, CSE, Chandigarh Engineering College, Landran, India
Astha Gupta , IT, Chandigarh Engineering College, Landran, India.

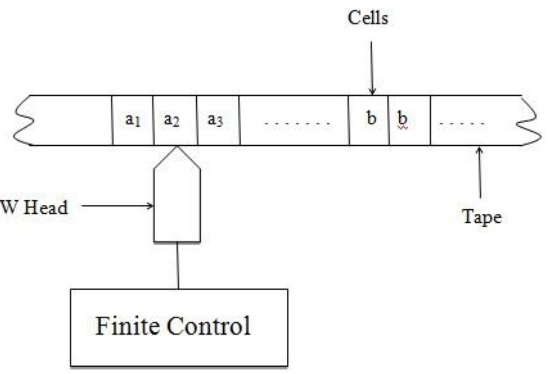

Figure 1.1: Turing machine

The Church-Turing thesis states that any algorithmic methodology that can be done by individuals/PC can be done by a Turing machine. It has been all around acknowledged by PC researchers that the Turing machine gives a perfect hypothetical model of a PC [7] [18].

Turing machines are very valuable in a few different ways. As a robot, the Turing machine is the most broad model. It acknowledges type-0 dialects. It can likewise be utilized for figuring capacities. It ends up being a scientific model of fractional recursive capacities. Turing machines are additionally utilized for deciding the un-decidability of specific dialects and estimating the reality multifaceted nature of issues. In Turing machines, the worthiness of a string is chosen by the reachability from the underlying state to some last state. So the last states are additionally called the tolerant states [18].

D. *Busy Beaver*

Assume a Turing Machine (TM) with a two way boundless tape and a tape letter set = {blank, 1} (the image 0 is utilized as clear image) [2]. Likewise expect that Turing machine at first totally clear and the machine must move either left or ideal at each progression, i.e., it can't stay stationary. There is single stopping state from which no changes develop, and this end state isn't included in all out number of states [19]. The inquiry Rado posed was: What is the most extreme number of 1's (not really back to back) that can be composed on the tape after such a N-state Turing machine stops, when begun a clear tape [2] [19]? This number, which is capacity of the quantity of states, is meant by ∑N (Rado's capacity). A machine that produces ∑N non-clear cells is known as a Busy Beaver (BB) [2]. Rado additionally characterized a capacity S (N) which tallies the most extreme number of moves that can be made by N-state ending Turing machine of this structure [19]. In the event that a Turing machine composes the greatest conceivable number of 1's for its number of states then it is known as a "Bustling Beaver" [19].

Occupied Beaver are elusive, notwithstanding for moderately little n, for two reasons. In the first place, the hunt space is very huge – there are (4(N+1))2N diverse Turing machines with N-states. Second, it is when all is said in done unrealistic to decide if a specific Turing machine will end or not. Thus, plainly neither ∑N nor S(N) are processable capacities – it implies that there is no ending Turing machine, which on self-assertive info N, will constantly stop and effectively figure these capacities [19].

∑N develops exceptionally quicker than any processable capacity [2] [11]. In any case, it is conceivable to process ∑N and S(n) for little estimations of N [19]. As the quantity of states builds, the issue ends up more enthusiastically. There is no specific hypothesis about the structure of Busy Beaver. The main route for discovering such machines is to playing out a thorough quest for all N-state Turing machines [2] [19] . The following table gives what is known about ∑N and S(N) $1 \le N \le 5$ [19].

II. PROPOSED WORK

Theory is relevant to practice. Computational complexity is an important branch of the theory of computation. It aims to tell us how many resources (times or spaces) a problem is solved with. The research on computational complexity could help us understand capabilities and limitations of the computer much better. If a problem is proved to be solved with great difficulty, people need not spend much effort on looking for efficient solving algorithm of the problem. Busy Beaver is basically a problem of Turing machines.
In [1], there are some results known to theoretically link some complexity notations, especially descriptional and computational complexity. This paper proposed that set of average run-time slows down by increasing the descriptional (program-size) complexity. The affect on the computational time complexity by increasing the number of states as a mean for increasing the program size (descriptional) complexity is studied. It is observed that by increasing the descriptional complexity (number of states), the number of algorithms computing less efficiently. In this paper, number of colors to k=2 are fixed. Number of states are increased as a mean for increasing the program-size (descriptional) complexity of the Turing machines in order to study any possible trade-offs with any of the other complexity measures, i.e., computational complexity.

To be more concrete, in this paper, TMs with 2 states and 2 colors are compared to TMs with 3 states and 2 colors. The main focus is on the functions they compute and the runtimes for these functions 4. Some of the questions we try to answer include what kind of, and how many functions are computed in each space? What kind of runtimes and space-usage do we typically see and how are they arranged over the TM space?
So, major purpose of this research work is to undertake an experimental investigation for examine the variation between the descriptional and computational time complexity for 5-state Busy Beaver function. It is already known that busy beaver is non-computable function. As like of Turing machines, number of states of Busy Beaver are also increased as a mean of increasing the descriptional complexity in order to study the effect on computational complexity. This experiment is performed for different N-values on different machines with different configuration and different platform.

Machines	Processor	RAM	Operating system
M_1	I5-3210M	6 GB	Linux – Ubuntu
M_2	Pentium(R) G3220	2 GB	Linux – Ubuntu
M_3	I3-350M	3 GB	Linux – Ubuntu
M_4	Pentium 4	2 GB	Linux- Ubuntu
M_5	I7 – 3220M	4 GB	Linux- Ubuntu

This experiment would help to understand the variation between the descriptional and computational time complexity on diverse machine configuration with various platforms. It will also help to understand whether the Busy Beaver functionis machine dependent or not. The machine dependency of Busy Beaver function is analysed by collecting the result of different machines with different configuration and different platform. A systematic and exhaustive study for analysis of runtime complexity for 5 state Busy Beaver function will be undertaken using an experimental setup and results should be analysed on some predetermined measures discussed above.

A. Methodology

N	∑N	S(N)
1	1	1
2	4	6
3	6	21
4	13	107
5	≥ 4098	≥ 47,276,870

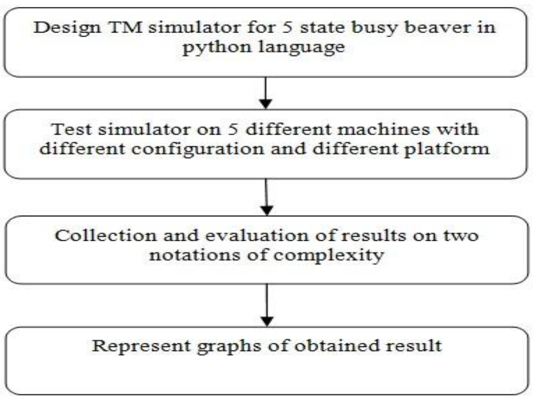

Figure 3.1: Flowchart of proposed work

B. Designing of TM Simulator

It is already known that Busy Beaver are hard to find. ∑N grows very faster than any computable function. Nevertheless, it is possible to compute ∑N and S (n) for small values of N. As the number of states increases, the problem becomes harder. So, a TM simulator will be designed for 5-state Busy Beaver function using Python programming language. TM simulator have programmed in python programming language to explore the different spaces of 5-state Busy Beaver function.

C. Testing of TM Simulator

Table 3.1: 5 different machines to test TM simulator

The designed simulator will be tested for different N-values on different machines with different configuration and different platforms. This simulator is tested on 5 different machines to check whether the Busy Beaver function is machine dependent or not.

D. Collection of results

The result is collected and evaluated on some basic notations of complexity, i.e, computational complexity and descriptional (program-size complexity). This systematic and exhaustive study for analysis of runtime complexity for 5 state Busy Beaver function will be undertaken using an experimental setup and results should be analysed on some predetermined measures discussed above. In particular it examine the time they take to compute in each space. The average runtime is collected on all different machines for each state. With every run at each state, the TM simulator calculated the three times namely; Real time, User time and System time.

One of these things is not like the other. Real time refers to actual elapsed time, User and System time refer to CPU time used *only by the process.*

Real time is wall clock time - time from start to finish of the call. This is all elapsed time including time slices used by other processes and time the process spends blocked (for example if it is waiting for I/O to complete).

User time is the amount of CPU time spent in user-mode code (outside the kernel) *within* the process. This is only actual CPU time used in executing the process. Other processes and time the process spends blocked do not count towards this figure.

System time is the amount of CPU time spent in the kernel within the process. This means executing CPU time spent in system calls *within the kernel,* as opposed to library code, which is still running in user-space. Like 'user', this is only CPU time used by the process.

The collected results are compared to understand whether the Busy Beaver function is machine dependent or not. This study will also reveal the effect on the run-time complexity by increasing the descriptional (program-size) complexity. The average run-time for computing the Busy Beaver function at one state is compared with the average run-time of the computing the Busy Beaver function at another state, which revealed the effect of descriptional complexity on computational complexity.

E. Graphical representation

Pictures speak louder than words. So, the obtained results are represented graphically on the basis of two parameters. These parameters are number of runs and calculated run time of the busy beaver function at every state. The collected results of all the machines at each state and at each run are evaluated and visualized into graphs, which will explain the behavior of busy beaver function on different machines with different configuration and different platform. It is also helpful in understanding the effect on computational complexity with increase in the descriptional complexity.

III. RESULTS AND DISCUSSION

A. Results

The designed simulator is tested on 5 different machines with different configuration and different platform. The simulator is tested for 10 times at each state. It will give the following results.

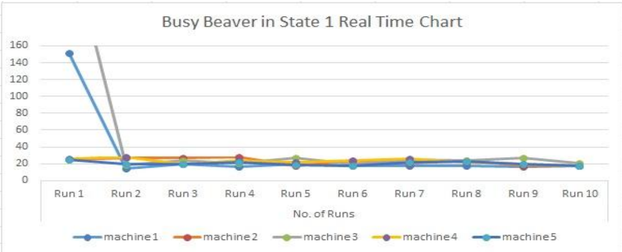

Figure 3.1: Real time chart on state 1

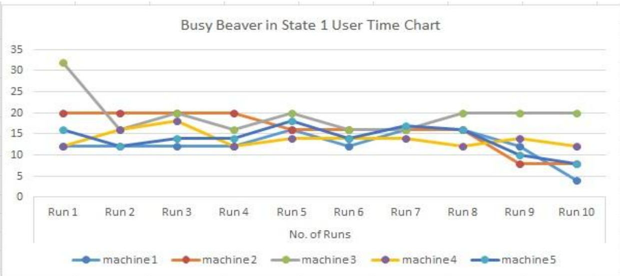

Figure 3.2: User time chart on state 1

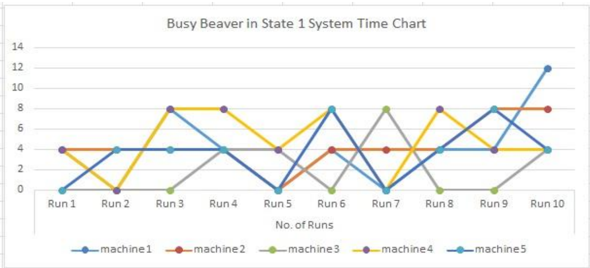

Figure 3.3: System time chart on state 1

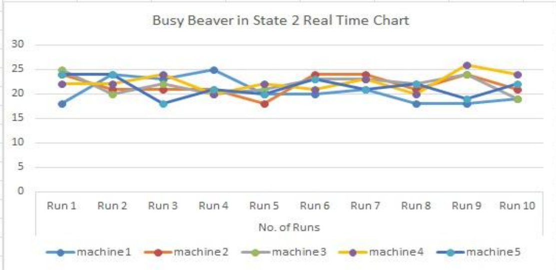

Figure 3.4: Real time chart on state 2

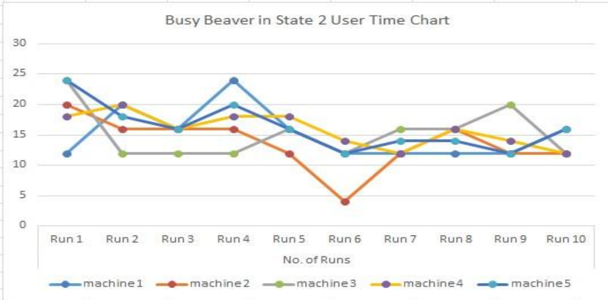

Figure 3.5: User time chart on state 2

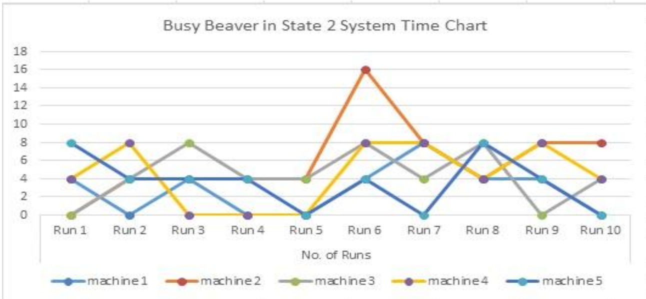

Figure 3.6: System time chart on state 2

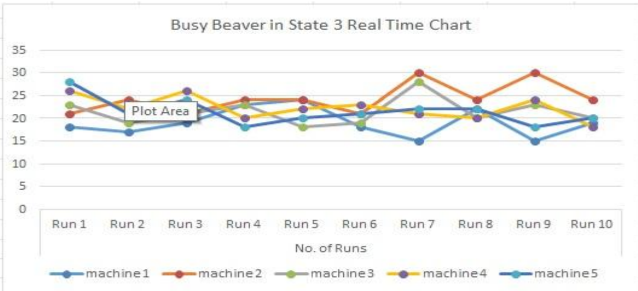

Figure 3.7: Real time chart on state 3

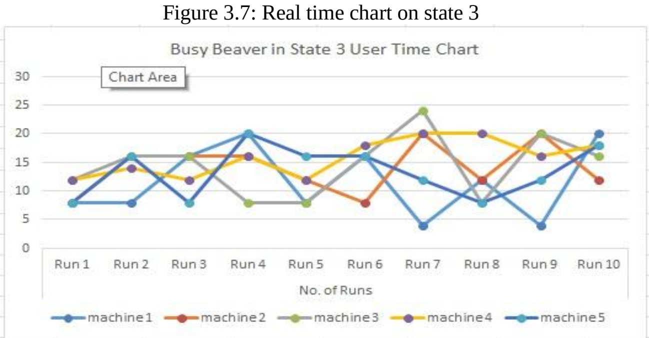

Figure 3.8: User time chart on state 3

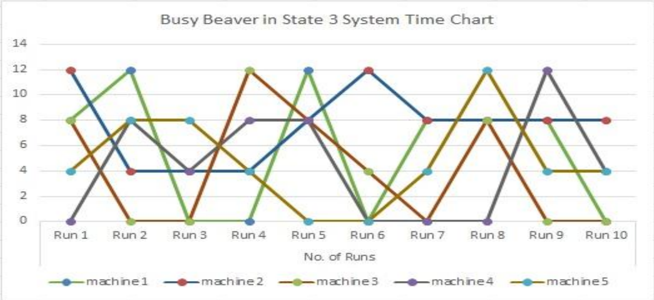

Figure 3.9: System time chart on state 3

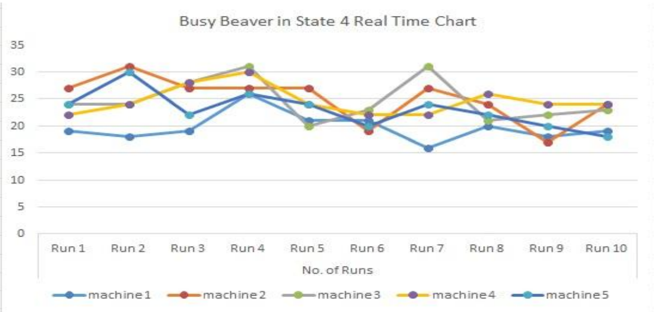

Figure 3.10: Real time chart on state 4

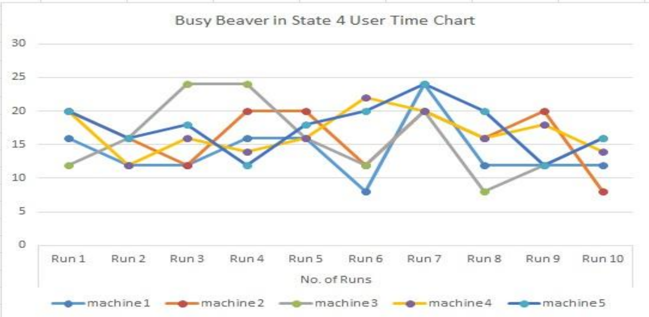

Figure 3.11: User time chart on state 4

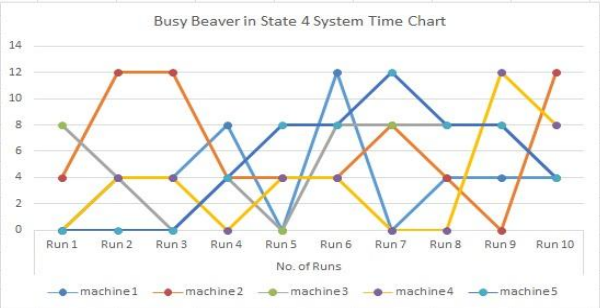

Figure 3.12: System time chart on state 4

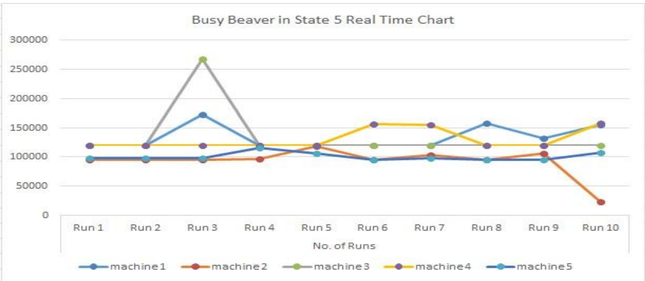

Figure 3.13: Real time chart on state 5

Table 4.1: Average Run time complexity on Machine1

State	Average Run-time	Average User time	Average System time
1	0m0.0312s	0m0.0124s	0m0.004s
2	0m0.0206s	0m0.0152s	0m0.0028s
3	0m0.019s	0m0.0116s	0m0.0056s
3	0m0.0207s	0m0.014s	0m0.004s
4	0m0.1334148 s	0m0.379516a	0m0.5604s

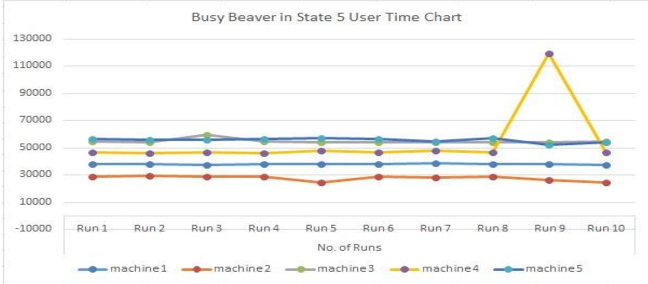

Figure 3.14: User time chart on state 5

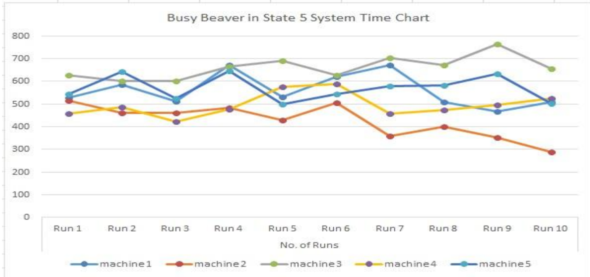

Figure 3.15: System time chart on state 5

Now, from the above results the average time for the 5 different machines will be:

Table 4.2: Average Run time complexity on Machine2

State	Average Run-time	Average User time	Average System time
1	0m0.023s	0m0.016s	0m0.0044s
2	0m0.0219s	0m0.0137s	0m0.006s
3	0m0.0243s	0m0.014s	0m0.0076s
4	0m0.025s	0m0.0164s	0m0.0064s
5	1m0.1024187s	0m0.277756s	0m0.4253s

Table 4.3: Average Run time complexity on Machine3

Sr No	Average Run-time	Average User time	Average System time
1	0m0.0495s	0m0.0196s	0m0.002s
2	0m0.0219s	0m0.0152s	0m0.0044s
3	0m0.0214s	0m0.0144s	0m0.04s
4	0m0.0247s	0m0.016s	0m0.0052s
5	0m0.1341793s	0m0.548632s	0m0.6608s

Table 4.4: Average Run time complexity on Machine4

State	Average Run-time	Average User time	Average System time
1	0m0.0765s	0m0.0139s	0m0.0036s
2	0m0.0226s	0m0.040s	0m0.0044s
3	0m0.0755s	0m0.0136s	0m0.0038s
4	0m0.0224s	0m0.0144s	0m0.0044s
5	0m0.7097984s	0m0.557955s	0m0.5695s

Table 4.5: Average Run time complexity on Machine5

State	Average Run-time	Average User time	Average System time
1	0m0.023s	0m0.138s	0m0.0048s
2	0m0.0224s	0m0.0158s	0m0.0044s
3	0m0.02222s	0m0.0158s	0m0.0040s
4	0m0.0223s	0m0.0138s	0m0.0044s
5	0m0.0653409s	0m0.430856s	0m0.4954s

IV. CONCLUSIONS & FUTURE SCOPE

A systematic and exhaustive study is undertaken for 5-state busy beaver function. For larger number of states, results are yet to be interrupted. Busy Beaver is basically a problem of Turing machine. There are some functions, which are not Turing computable. A lot of efforts are undertaken to calculate the values of non-computable Busy Beaver function. It is really quite fascinating to contemplate the successful efforts which have been made to calculate some of the initial values of ∑N. In this research work, analyzing the data gave us interesting functions with their geometrical patterns in terms of run time and descriptional complexity. The average run time of computing a function slows down with increase in the descriptional complexity because picking and algorithm at random from the number of algorithms of computing a function in large number of states leads to greater chance to pick slow algorithm as compared to number of fastest algorithm in same space. One made an additional effort in order to pick a faster algorithm without having to spend larger and larger resources in the search of an efficient algorithm.

The geometrical charts revealed that the busy beaver function is only machine-dependent when it is tested on the machine in the kernel with in the process, i.e., only the system time is dependent on the configuration of machine. It changes with the change in the configuration of machine. The real time and user time almost remain same with the change in the configuration and platform of machine.

In future work, the efforts can be made to calculate the ∑N for large value of N. Secondly, the search space is extremely large. There are (4(N+1))2N different Turing machines with N-state. So, busy beaver are hard to find. It is difficult to find whether a particular TM will halt or not. So, the efforts can be made to determine whether a particular TM will halt or not.

REFRENCES

1. Joost J. Joosten, Fernando Soler-Toscano and Hector Zenil "Program-size versus Time complexity Slowdown and speed-up phenomena in the micro-cosmos of small Turing machines," 16 April 2011.
2. Francisco B. Pereira, Penousal Machado, Ernesto Costa, Amílcar Cardoso, "Busy Beaver – An Evolutionary Approach"
3. Qiang Gao and Xu Xinhe "The analysis and research on computational complexity," Control and Decision Conference, The 26th Chinese, IEEE 2014.
4. Claus Diem, "On the complexity of some computational problems in the Turing model," Preprint, November 18, 2013.
5. Cristian S. Calude, Michael A. Stay, "Most programs stop quickly or never halt," Advances in Applied Mathematics 40, (2008).
6. Gregory J. Chaitin, "Computing the Busy beaver function,"In T. M. Cover and B. Gopinath, Open Problems in Communication and Computation, Springer, pp. 108–112,1987.
7. Turlough Neary, Damien Wood, "Small fast universal Turing machines," Theoretical Computer Science 362, 1 June 2006.
8. Pascal Michel, "Small Turing machines and generalized busy beaver competition," Theoretical Computer 326, 18 May 2004.
9. Marxen, Heiner,Specs Gmbh, "Attacking the busy beaver 5," Bull EATCS. 1990.
10. Woods Damien, and Turlough Neary, "The complexity of small universal Turing machines: A survey," Theoretical Computer Science 410.4, 2009.
11. Francisco B Pereira, et al. "Graph based crossover–a case study with the busy beaver problem," Proceedings of the 1999 Genetic and Evolutionary Computation Conference, 1999.
12. Ed Blakey, "Computational Complexity in Non-Turing Models of Computation: The What, the Why and the How," Electronic Notes in Theoretical Computer Science 270.1, 2011.
13. Thomas Worsch, "Parallel Turing machines with one head control units and cellular automata," Theoretical computer science 217.1,1999.
14. Jones, Terry, and Gregory JE Rawlins, "Reverse Hill climbing Genetic Algorithms and the Busy Beaver Problem," ICGA, 1993.
15. Rado T., "On non-computable functions," The Bell System Technical Journal, vol. 41, no. 3, pp.877-884, 1962.
16. Brady, A.H., "The conjectured highest scoring machines for Rado's S(k) for the value k=4," IEEE Transactions on Electronic Computers, vol. EC-15, pp. 802-803, October 1966.
17. Brady, A.H., "Solution of the non-computable Busy Beaver game for k=4," Abstracts for ACM Computer Science Conference Washington, DC, p. 27, ACM, February 18-20, 1975.
18. K.L.P Mishra, N. Chandrasekaran, "Theory of computer science," Third edition.
19. University of Waterloo, "Introduction to the theory of computing – Handout on the Busy Beaver problem," Winter, 1998.
20. K. Dewdney, "A computer trap for the busy beaver, the hardest-working Turing machine," Computer Recreations Dept., Scientific American 251, No. 2, Aug, 1984.
21. Penousal Machado, Francisco B. Pereira, Amílcar Cardoso , Ernesto Costa, " Busy Beaver – The Influence of Representation".

Retrieval Number: I11170789S19/19©BEIESP
DOI: 10.35940/ijitee.I1117.0789S19

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

AUTHORS PROFILE

Pardeep Singh Tiwanareceived the B.tech and M.tech degrees in Information technology and engineering from Punjab Technical University, Jalandhar, India, in 2014 and 2016 respectively. He is working towards the Ph.D degree with Chandigarh University India and also working as Assistant Professor in Chandigarh Engineering College.

Harjot Singh Tiwanareceived the B.tech and M.tech degrees in CSE and engineering from Punjab Technical University, Jalandhar, India, in 2013 and 2015 respectively. He is working towards the Ph.D degree with Chandigarh University India and also working as Assistant Professor in Chandigarh Engineering College.

Rajeev Sharmareceived the B.tech and M.tech degrees in CSE and engineering from Punjab Technical University, Jalandhar, India. He is working towards the Ph.D degree with Chandigarh University India and also working as Assistant Professor in Chandigarh Engineering College.

Astha Guptareceived the B.tech and M.tech degrees in Information technology and engineering from Punjab Technical University, Jalandhar, India. He is working towards the Ph.D degree with Chandigarh University Indiaand also working as Assistant Professor in Chandigarh Engineering College.