International Journal of Enhanced Research in Management & Computer Applications
ISSN: 2319-7471, Vol. 5 Issue 6, June-2016, Impact Factor: 1.544

A Tentative Set of Connections to Investigate 5-State Busy Beaver Using Run-Time Complexity

Komal Singla[1], Sunil Maggu[2]

[1,2] Department of Computer Science and Engineering, Vaish College of Engineering, Rohtak, India

ABSTRACT

The significant point of this paper is to embrace an exploratory examination for break down the change between the computational complexity and discretional complexity nature. To endure out exploration, Turing Machine simulator for Busy Beaver function will be weathered for dissimilar N-values on dissimilar machines with different game plan and various propositions to work out the run-time complexity nature. This learning encourage whether the Busy Beaver function is machine dependent. It besides accounted with the aim of the average run-time of Busy Beaver work unquestionably increments as the number of states.

Keywords: Busy Beaver function, Computational complexity, Descriptional complexity, Turing Machine

I. INTRODUCTION

At this time there are excess of techniques toward surveying the computational complexity in spite of the fact that various them concentrating on top of evaluating the benefits much the same as moment, crevice and force worn by method for count. The key inspiration driving the examination is to make out the bond flanked by the complexity exercises, particularly the way of computational complexity and discretional complexity. It has been deliberately clarified in the underneath area.

A. Computational Complexity

Computational complexity is a range office of the theory of calculations. Computational complexity of the situation is the way various strides it takes to unwind the bind by means of the biggest piece of triumphant calculation.

B. Descriptional Complexity

Descriptional Complexity of a twofold arrangement is termed the same as the ostensible plan to generate the arrangement. Close by there is no obvious method which creates the undeviating calculation with the point of delivering a prearranged arrangement.

C. Turing Machine

Turing machine can work out everything, which is assessable [3]. Turing machine has two way interminable tapes which is isolated into number of cells. Cell can be a non clear image or can be a clear. All cell contains only one image. Turing machine has one head, known as R\W head (Read and Write head) that move over the cells of tape. R/W head can analyze the one cell at once. At every 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 [3] [4].

1) Definition: A Turing machine M has 7 tuple specifically (Q, $\sum$, Γ, ∂, q_0, b, F,) where

- Q is a limited non void arrangement of states.
- $\sum$ is a non unfilled arrangement of information images and is a subset of Γ.
- Γ is a limited non void arrangement of tape images.
- ∂ is move capacity mapping (q, x) onto (q, y, D) where D is bearing of development of R/W head.
- $q_0 \in Q$ is the underlying state and
- $b \in \Gamma$ is clear.
- $F \subseteq Q$ is arrangement of definite states [3] [4].

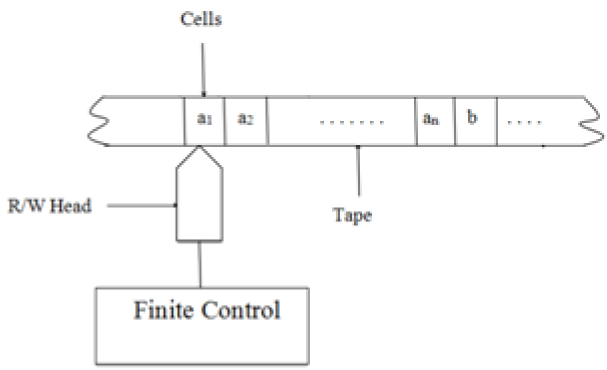

Fig. 1 Turing Machine

It has been all around recognized by PC researchers that the Turing machine gives an extreme hypothetical model of a PC. In Turing machines, the ampleness of an arrangement is resolved through reach ability from the underlying state to some last state. So the last states are likewise called the tolerant states [13].

D. Busy Beaver

Assume a Turing Machine (TM) with a two way endless tape and a tape letters in order = {blank, 1} (the image 0 is utilized as clear image) [2].Additionally accept that Turing machine at first totally clear and the machine must move either left or comfortable stride, i.e., it can't stay stationary. There is single stopping state from which no moves develop, and this end state is not tallied in complete number of states [14]. Busy Beaver are elusive, notwithstanding for moderately little n as there are (4(N+1))2N distinctive Turing machines with N-states. S(N) tallies the greatest number of moves that can be made by N-state stopping Turing machine of this structure . A machine that produces ∑N non-clear cells is known as a Busy Beaver (BB) [14].

Key standard of this examination is to tackle a speculative investigation intended for taking a gander at the inconsistency flanked by the descriptional and computational time complexity for 5-state Busy Beaver function. A portion of the inquiries we attempt to answer incorporate what sort of, and what number of capacities are registered in every space? What sort of runtimes and space-use do we ordinarily see and how are they orchestrated over the TM space?

II. PROBLEM FORMULATION

In [3], there are some outcomes recognized to speculatively interface different intricacy documentations, especially descriptional and computational complexity. This paper arranged with the point of normal run-time complexity. It diminishes by expanding the descriptional complexity .The method for the computational time complexity nature by raising the level of states as a mean for creating descriptional complexity nature is figured. It is handy that by rising the descriptional complexity nature (number of states), the quantity of calculations registering less effectively. In this paper, number of colours to k=2 are fixed. Number of states are puffed-up as a mean for rising the descriptional complexity of the Turing machines in course to take in any conceivable exchange offs with a few of the past intricacy measures, i.e., computational complexity. To be more concrete, in this paper, TMs with 2 states and 2 colours are contrasted with TMs with 3 states and 2 colours. The fundamental center is on the functions they figure and the runtime for these functions.

Along these lines, key standard of this examination is to tackle a speculative investigation intended for taking a gander at the inconsistency flanked by the descriptional and computational time complexity for 5-state Busy Beaver function. It is at this point perceived that busy beaver is non-calculable function. As like Turing machines, number of conditions of Busy Beaver is additionally broadened as a sign of heightening the descriptional complexity in course of contemplating the result of computational complexity. This testing is intended for divergent N-values on dissimilar machines with different game plan and disparate proposition. This testing would see the refinement flanked by the descriptional and computational time complexity nature on dissimilar machine advancement through a variety of stages. It will furthermore help us to perceive whether the Busy Beaver capacity is machine dependent or not. The machine importance of Busy Beaver function is broke down by gathering the outcome dissimilar machines with unique game plan and different proposition. A systematic and extensive learning for examination of runtime complexity nature for 5-state Busy Beaver function will be attempted by tentative set of connections.

III. METHODOLOGY

1) Step 1: Plan TM simulator for 5 state Busy Beaver in python language.

2) Step 2: Examine simulator on 5 dissimilar machines with different game plan and disparate proposition.
3) Step 3: Gathering and assessment of consequences on two notations of complexity.
4) Step 4: Representing graphs of obtained grades.

IV. TESTING OF TM SIMULATOR

The expected test system will be weathered for dissimilar N-values on dissimilar machines with different game plan and various proposition. This test system is weathered on 5 divergent machines to ensure whether the Busy Beaver capacity is gadget subordinate or not. The 5 dissimilar machines of unique course of action and various proposition are talked about beneath.

Table I: Different Machines To Test Tm Simulator

Machines	Processor	RAM	Operating System
M1	I3-2Ghz	4 GB	Linux-Ubuntu
M2	Dual Core	3 GB	Linux-Ubuntu
M3	I5-3210M	6GB	Linux-Ubuntu
M_4	Pentium 4	2 GB	Linux-Ubuntu
M_5	I7 – 3220M	4 GB	Linux-Ubuntu

The TM test system is weathered on first machine (M1). Test system is weathered for 10 times at each circumstance. Along these lines, we can say that TM test system is experienced for 10 times at state 1, taking after that the state is augmented and test system is again weathered for 10 times at state 2.

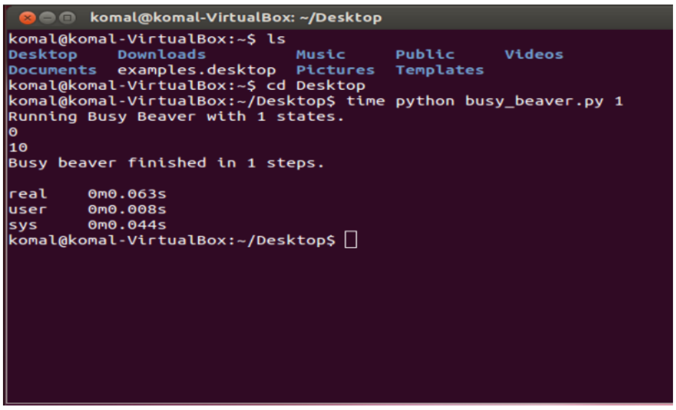

Fig. 1 Testing the busy beaver simulator on state 1

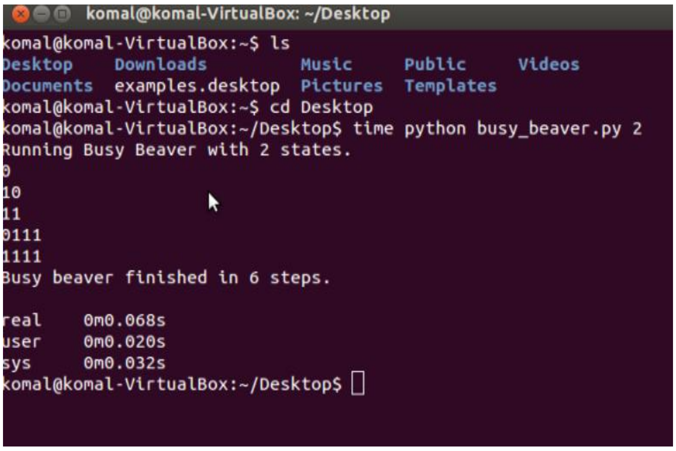

Fig. 2 Testing busy beaver simulator in state 2

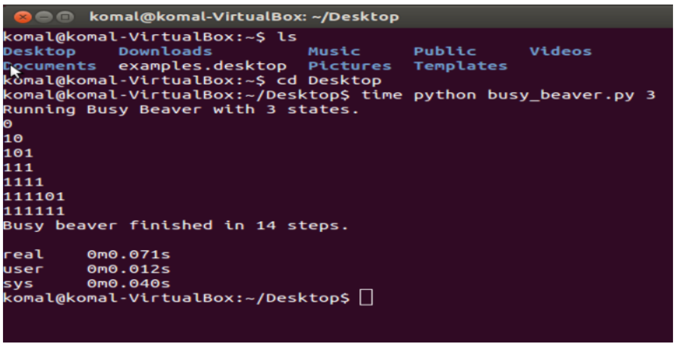

Fig. 3 Testing busy beaver simulator in state 3

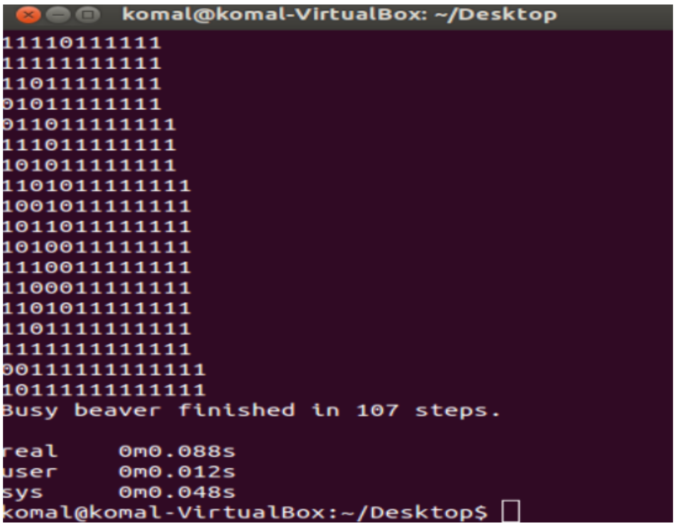

Fig. 4 Testing busy beaver simulator in state 4

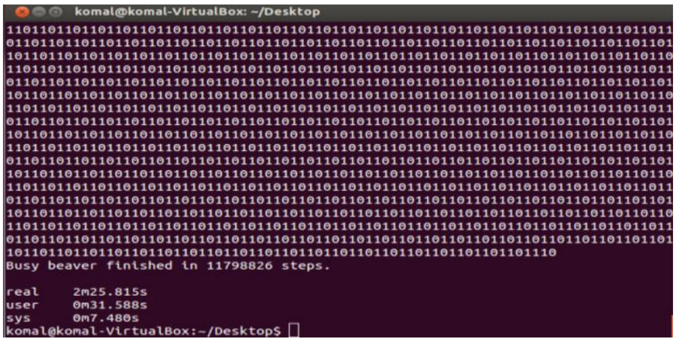

Fig. 5 Testing busy beaver simulator in state 5

V. RESULTS AND DISCUSSIONS

The result is assembled and surveyed on some central documentations of complexity, i.e., computational complexity and descriptional complexity. . In basic words it ponder the time they take to work out in every space. The normal runtime is assembled on differing machines for all state. By every keep running at each state, the TM test system proposed for three times in particular; Real time, User time and System time.

One of this stuff is not much the same as the other. Continuous alludes to unmistakable over and done time; User and system time allude to CPU time worn essentially by the procedure.

1) Real Time: It is divider clock time means time starting from begins to end of the call. This is all over and done time and additionally time cuts utilized by previous procedures and time the procedures spend stuck.
2) User Time: User Time is the measure of CPU time exhausted in client mode code encompassed by the procedures. This is just clear CPU time worn in executing the procedure.
3) System Time: It is the measure of CPU time exhausted in the bit encompassed by the procedure.

Now the designed simulator is weathered on 5 dissimilar machines with dissimilar arrangement and diverse proposals. The simulator is weathered for 10 times at every state. It will give the following results.

Machines			Machine1	Machine 2	Machine 3	Machine 4	Machine 5
Real time	No. of Runs	Run 1	62	43	152	26	20
		Run 2	61	32	14	28	25
		Run 3	72	29	20	20	20
		Run 4	75	28	20	24	22
		Run 5	0	28	17	22	19
		Run 6	73	29	18	26	18
		Run 7	74	28	18	24	22
		Run 8	71	41	18	20	23
		Run 9	77	41	17	18	20
		Run 10	63	40	18	20	18

Fig.6 Real time on state 1

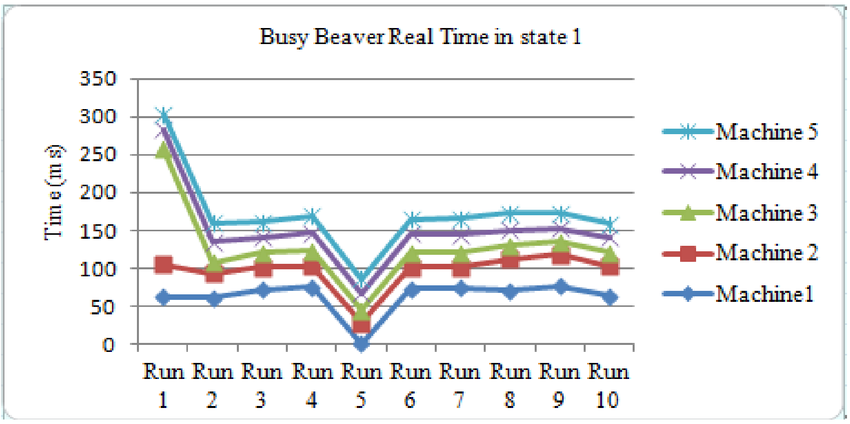

Fig. 7 Real time chart on state 1

Machines			Machine1	Machine 2	Machine 3	Machine 4	Machine 5
User Time	No. of Runs	Run 1	16	21	0	4	0
		Run 2	16	17	0	0	4
		Run 3	12	21	0	8	4
		Run 4	16	17	4	8	4
		Run 5	20	20	4	4	0
		Run 6	16	21	0	8	8
		Run 7	20	12	8	0	0
		Run 8	24	25	0	8	4
		Run 9	16	13	0	4	8
		Run 10	8	22	4	4	4

Fig. 8 User Time on state 1

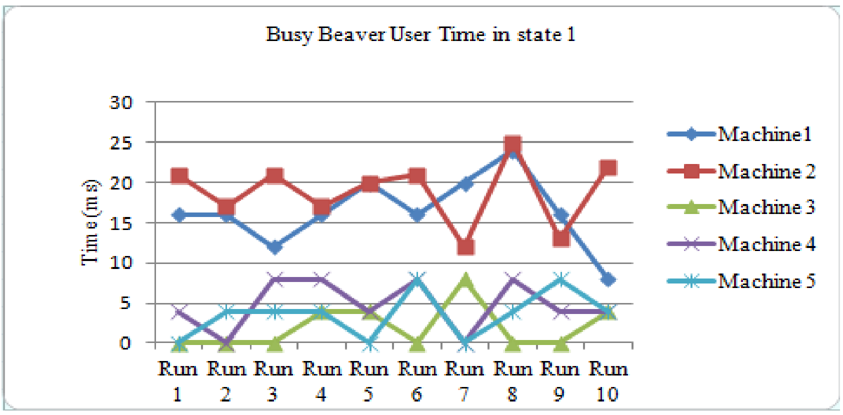

Fig. 9 User time chart on state 1

Machines			Machine1	Machine 2	Machine 3	Machine 4	Machine 5
System Time	No. of Runs	Run 1	32	7	0	4	0
		Run 2	36	8	0	0	4
		Run 3	32	4	0	8	4
		Run 4	36	4	4	8	4
		Run 5	36	8	4	4	0
		Run 6	32	4	0	8	8
		Run 7	24	4	8	0	0
		Run 8	32	12	0	8	4
		Run 9	44	0	0	4	8
		Run 10	32	4	4	4	4

Fig. 10 System time on state 1

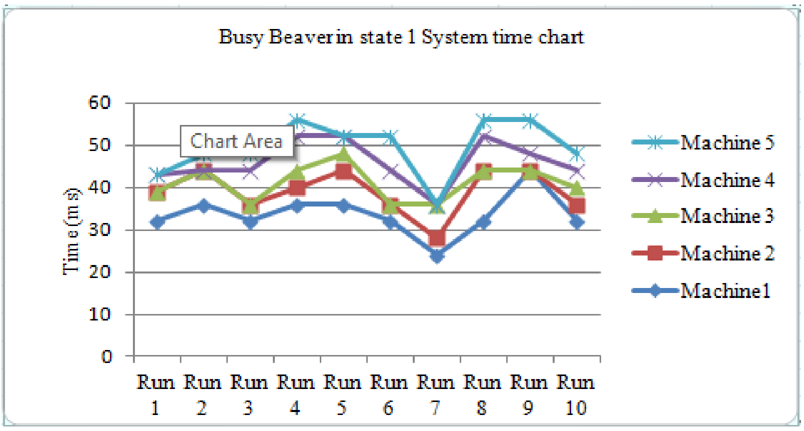

Fig. 11 System time chart on state 1

Machines			Machine1	Machine 2	Machine 3	Machine 4	Machine 5
User Time	No. of Runs	Run 1	12	11	12	12	16
		Run 2	24	18	20	14	12
		Run 3	4	21	16	16	14
		Run 4	20	17	24	12	14
		Run 5	16	22	16	14	12
		Run 6	8	21	12	18	16
		Run 7	20	21	12	18	20
		Run 8	8	20	12	16	16
		Run 9	4	0	12	20	18
		Run 10	20	7	16	18	24

Fig. 12 User time on state 2

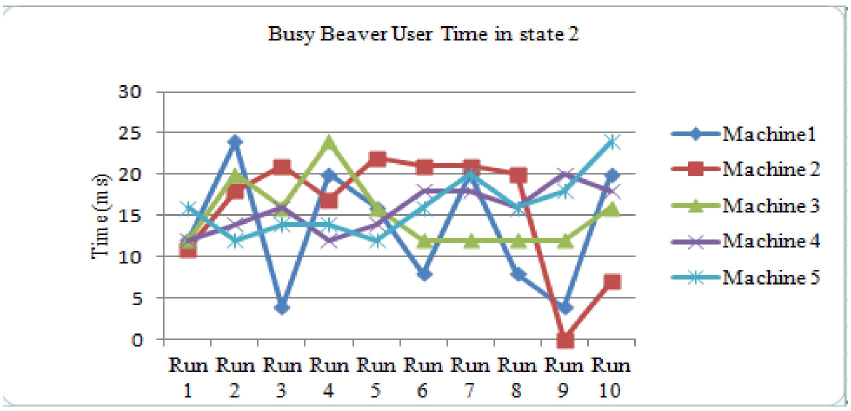

Fig. 13 User time chart on state 2

Machines			Machine1	Machine 2	Machine 3	Machine 4	Machine 5
Real time	No. of Runs	Run 1	80	38	24	22	24
		Run 2	73	40	18	22	24
		Run 3	105	32	23	24	18
		Run 4	78	44	19	20	21
		Run 5	72	38	18	22	20
		Run 6	115	46	18	21	23
		Run 7	77	42	21	23	21
		Run 8	76	38	20	20	22
		Run 9	124	32	20	26	19
		Run 10	79	40	25	24	22

Fig. 14 Real time on state 2

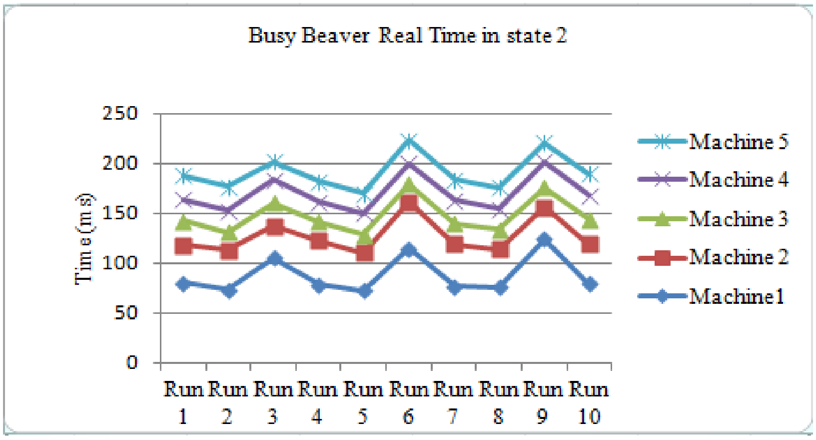

Fig. 15 Real time chart on state 2

Machines			Machine1	Machine 2	Machine 3	Machine 4	Machine 5
System Time	No. of Runs	Run 1	44	14	4	0	4
		Run 2	28	7	0	8	8
		Run 3	80	4	4	0	4
		Run 4	32	17	0	4	4
		Run 5	36	4	0	0	0
		Run 6	88	4	4	8	0
		Run 7	36	3	8	8	8
		Run 8	48	4	4	4	4
		Run 9	36	18	4	8	4
		Run 10	88	7	0	4	0

Fig. 16 System time on state 2

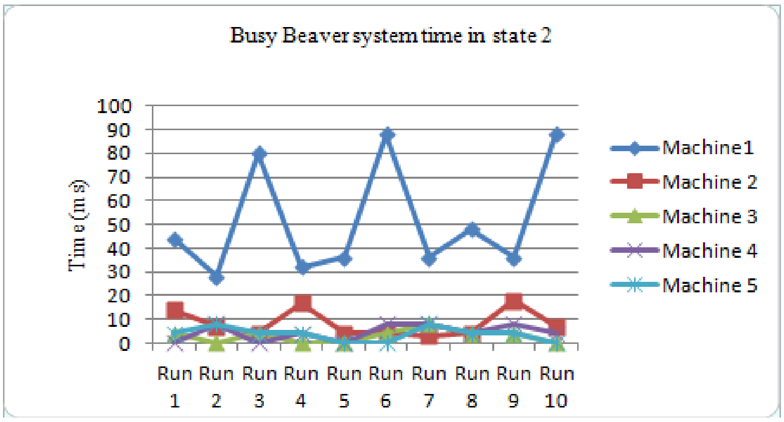

Fig. 17 System time chart on state 2

Machines			Machine1	Machine 2	Machine 3	Machine 4	Machine 5
Real time	No. of Runs	Run 1	145815	150299	119543	119545	95745
		Run 2	145681	167371	119673	119675	95574
		Run 3	193819	167456	171866	119666	97463
		Run 4	145108	168621	119433	119876	97435
		Run 5	147986	165001	119386	119124	97435
		Run 6	142670	168216	119855	156345	115645
		Run 7	147404	165550	119521	155231	106543
		Run 8	143579	164584	157660	119356	95463
		Run 9	147533	167002	132210	119255	97765
		Run 10	144036	166691	155001	157745	106986

Fig. 18 Real time on state 5

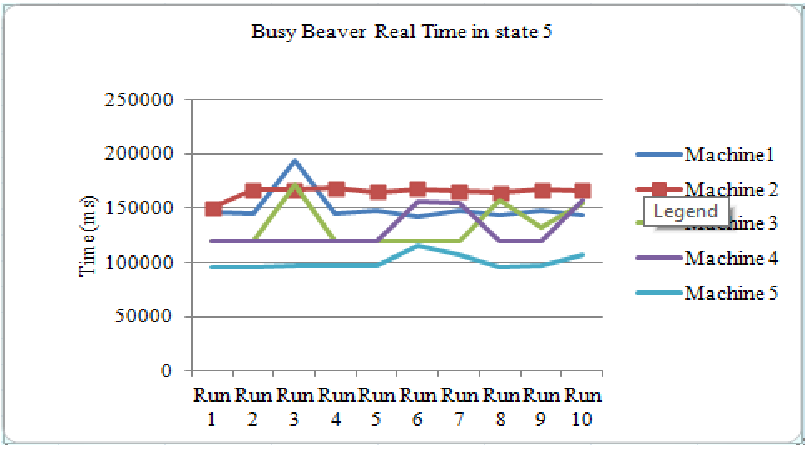

Fig. 19 Real time chart on state 5

Machines			Machine1	Machine 2	Machine 3	Machine 4	Machine 5
User Time	No. of Runs	Run 1	31588	47161	37928	46675	54214
		Run 2	31940	48025	37356	119346	52346
		Run 3	32260	46382	37888	46575	56975
		Run 4	31936	47248	37544	47634	54734
		Run 5	32528	47691	38136	46745	56573
		Run 6	31148	46712	37948	47654	56867
		Run 7	32044	46173	38728	46235	56676
		Run 8	31916	46439	38000	46653	56000
		Run 9	32656	47440	38176	46346	56234
		Run 10	32910	45890	37812	46634	56436

Fig. 20 User time on state 5

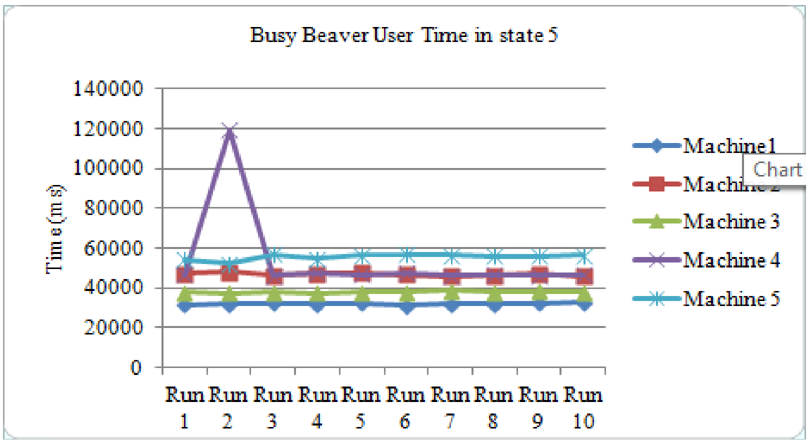

Fig. 21 User time chart on state 5

Machines			Machine1	Machine 2	Machine 3	Machine 4	Machine 5
System Time	No. of Runs	Run 1	480	657	408	424	502
		Run 2	456	697	568	597	634
		Run 3	784	634	508	465	583
		Run 4	144	664	672	472	578
		Run 5	332	729	620	465	543
		Run 6	724	650	532	468	500
		Run 7	724	666	672	423	645
		Run 8	724	679	512	487	524
		Run 9	724	688	584	578	634
		Run 10	724	703	528	456	534

Fig. 22 System time on state 5

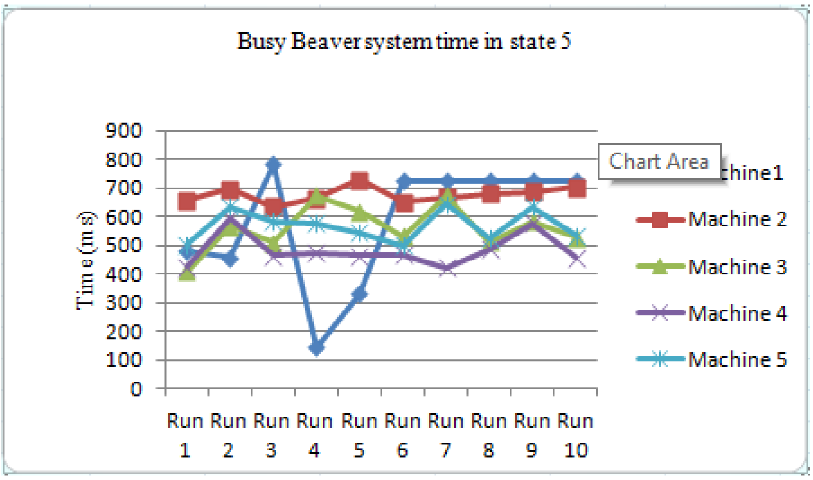

Fig. 23 System time chart on state 5

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

Table II: Average Run Time Complexity on Machine 1

State	Average Real Time	Average User Time	Average System Time
1	0m0.0691s	0m0.0164s	0m0.0344s
2	0m0.0879s	0m0.0136s	0m0.0516s
3	0m0.0757s	0m0.0188s	Om0.0310s
4	0m0.01074s	0m0.0176s	0m0.0416s
5	0m0.14985001s	0m0.242595s	0m0.4580s

Table III: Average run time complexity on machine 2

State	Average Real Time	Average User Time	Average System Time
1	0m0.0339s	0m0.0189s	0m0.0063s
2	0m0.0390s	0m0.0169s	0m0.0073s
3	0m0.0385s	0m0.0141s	0m0.0072s
4	0m0.0453s	0m0.0174s	0m0.0092s
5	0m0.1370791s	0m0.469161s	0m0.6767s

Table IV: Average run time complexity on machine 3

State	Average Real Time	Average User Time	Average System Time
1	0m0.0312s	0m0.0124s	0m0.004s
2	0m0.0206s	0m0.0152s	0m0.0028s
3	0m0.0019s	0m0.0116s	0m0.0056s

4	0m0.0207s	0m0.014s	0m0.004s
5	0m0.1334148s	0m0.379516s	0m0.5604s

Table V: Average run time complexity on machine 4

State	Average Real Time	Average User Time	Average System Time
1	0m0.0023s	0m0.0138s	0m0.0048s
2	0m0.0224s	0m0.0158s	0m0.0044s
3	0m0.0222s	0m0.0158s	0m0.0040s
4	0m0.0223s	0m0.0138s	0m0.0044s
5	0m0.0653409s	0m0.430856s	0m0.4954s

Table VI: Average run time complexity on machine 5

State	Average Real 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

Here key principle of this investigation is at first, the analysis is proceeded to examine the consequence on run time complexity with escalating the discretional complexity and the other one is the research is performed to look at whether the busy beaver function is device dependent or not. So, the run time of busy beaver is planned at all states. The dissimilarity in the run time is analyzed with every enlarge in the state of busy beaver. It is apparent from the average run time complexity tables that the average run time slows down by escalating the state of the busy beaver function. It is discovered that escalating the discretional complexity (number of states), the number of algorithms computing less professionally. In simple terms, it is obvious that the average run time of computing a function almost rises with increases in the number of states.

Research is performed on dissimilar machines with dissimilar arrangement and diverse proposals as well. Afterward, the consequences are represented graphically on the foundation of two parameters. So, it is apparent from the charts which are shown on top that user time, real time, system time all are machine dependent. It exposed that the system time is totally depended upon the arrangement of the machine. If the research is carried out on an additional machine with dissimilar arrangement and diverse proposal, the user time, real time, and system time will definitely alter. So the amount of CPU time worn-out in the kernel and outside the kernel varies with modifying the arrangement of machine.

CONCLUSIONS AND FUTURE WORK

A methodical and extensive learning is undertaken for 5-state busy beaver function. For a large number of states, consequences are so far to be interpreted. Busy Beaver is on the whole a quandary of Turing machine. There are various functions, which are not Turing computable. A lot of hard work is done to work out the standards of non-computable Busy Beaver function. It is in fact interesting to consider the hard work which has been done to work out a number of early values of $\sum N$. The average run time of figuring out a function decelerate by extending the descriptional complexity since selecting an algorithm casually from a number of algorithms working out a function in huge quantity of states show the way to better likelihood to select deliberate algorithm in contrasting to number of best ever algorithm in equivalent space.

The average run time of computing a function almost rises with increases in the number of states. The geometrical charts discovered that busy beaver function device-dependent when it is weathered on dissimilar machines with different game plan and disparate proposition. It alters with the variation in the arrangement of machine. In future work, the hard work can be done to work out the $\sum N$ for large value of N. Secondly, the investigation is extremely large. There are $(4(N+1))^{2N}$ unlike Turing machines with N-state. As a result, busy beaver functions are rigid to discover. It is tricky to come across whether a fastidious TM will halt or not. Accordingly, the hard work can be done to conclude whether a particular TM will halt or not.

REFERENCES

[1] Claus Diem, “On the complexity of some computational problems in the Turing model,” Preprint, November 18, 2013.

[2] Qiang Gao “The analysis and research on computational complexity,” Control and Decision Conference, The 26th Chinese, IEEE 2014.

[3] J. Joosten, “Program-size versus Time complexity Slowdown and speed-up phenomena in the micro-cosmos of small Turing machines,” 16 April 2011.

[4] Francisco B Pereira "Graph based crossover–a case study with the busy beaver problem," Proceedings of the 1999 Genetic and Evolutionary Computation Conference, 1999.

[5] Turlough Neary, “Small fast universal Turing machines,” Theoretical Computer Science 362, 1 June 2006.

[6] Penousal Machado, Amílcar Cardoso, “Busy Beaver – An Evolutionary Approach”

[7] 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.

[8] Cristian S. Calude, Michael A. Stay, “Most programs stop quickly or never halt,” Advances in Applied Mathematics 40, (2008).

[9] 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.

[10] Jones and Gregory JE Rawlins, "Reverse Hill climbing Genetic Algorithms and the Busy Beaver Problem," ICGA, 1993.

[11] Rado T., “On non-computable functions,” The Bell System Technical Journal, vol. 41, no. 3, pp.877-884, 1962.

[12] K. Dewdney, “A computer trap for the busy beaver, the hardest-working Turing machine," Computer Recreations Dept., Scientific American 251, No. 2, Aug, 1984.

[13] K.L.P Mishra “Theory of computer science,” Third edition.

[14] University of Waterloo, “Introduction to the theory of computing – Handout on the Busy Beaver problem,” Winter, 1998.

[15] Pascal Michel, “Small Turing machines and generalized busy beaver competition,” Theoretical Computer 326, 18 May 2004.

[16] Penousal Machado, Francisco B. Pereira, Amílcar Cardoso , Ernesto Costa, “ Busy Beaver – The Influence of Representation”.

[17] Woods Damien, and Turlough Neary, “The complexity of small universal Turing machines: A survey,” Theoretical Computer Science 410.4, 2009.

[18] Marxen, Heiner, Specs “Attacking the busy beaver 5,” Bull EATCS. 1990.