International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-2, July 2019

Parallel and Serial Graph Coloring Implementations with Tabu Search Method

Rutanshu Jhaveri, Narayanan Prasanth, Jayakumar K, Navaz K

Abstract: ***One of the well-known property of graph is graph coloring. Any two vertices of a graph are different colors such that they are adjacent to each other. The objective of this paper is to analyse the behavioral performance of Tabu Search method through serial and parallel implementations. We explore both parallel and serial Tabu search algorithm for graph coloring with arbitrary number of nodes.***

Index Terms: ***Algorithm analysis, Graph coloring, Parallel processing, Tabu search.***

I. INTRODUCTION

A graph is an illustrated representation of different sets of objects and the links between these objects. These links together give an abstract representation of relationships. In graph theory, the objects are called vertices, whereas the links in between them are called edges. A vertex is defined as an entity and on the other hand the edge, is said to be the relationship or association between these two entities. Vertex coloring is the most common graph coloring problem. The problem starts off simply, you have m colors and need to find a way to color the vertices of your graph in such a manner, where no two adjacent vertices connected by an edge have the same color. The minimum number of colors that one completes coloring a Graph G, is called the chromatic number. Other examples of graph coloring types include Edge Coloring – no vertex is incident to two edges, which have the same color, and Face Coloring – Geographical Map Coloring. However, we choose to avoid using these for our analysis as we are planning to work with vertices which are easier to calculate and visualize when speaking about a classroom arrangement or any such application. In addition, the aforementioned coloring techniques can be transformed into Vertex Coloring using various matrix operations. Various algorithms are used to find the Chromatic number for any Graph. They use different approaches and provide varying results depending on the graph size. In this paper, we evaluate the behavioral performance of Tabu Search method through serial and parallel implementations. The number of cores and graph size would be determining factors for each algorithm. The structure of the paper as follows: Section 2 presents the relative study of the work. Section 3 discusses the graph coloring algorithm and section 4 produces the implementation results and its discussion. Section 5 conclude the paper.

Revised Manuscript Received on 30 July 2019.

* Correspondence Author

Rutanshu Jhaveri, School of CSE, VIT University, Vellore, India.
Narayanan Prasanth, School of CSE, VIT University, Vellore, India.
K.Jayakumar, School of CSE, VIT University, Vellore, India.
K.Navaz, Dept. of CSE, Annamacharya Institute of Technology and Sciences Tirupathi, India

II. RELATED WORKS

Allwright et al. [1] presented some parallel graph coloring algorithms dependent on understood sequential heuristic calculations, and contrast them and some existing parallel algorithms. These calculations are actualized on both SIMD and MIMD parallel models and tried for speed, effectiveness, and for shading arbitrary triangulated networks and diagrams from sparse matrix. Buhua Chen et al. [2] introduced a new parallel genetic algorithm to take care of the Graph coloring problem (GCP) in view of Computer Unified Device Architecture (CUDA). All the operators such as initialization, crossover, mutation and selection are designed to be parallel in threads. Additionally, the execution of their algorithms is contrasted and alternate algorithms utilizing benchmark charts, and exploratory outcomes demonstrates that their calculation merges significantly more rapidly than different calculations and accomplishes focused execution for solving GCP. Frank et al. [3], the paper portrays another graph coloring algorithm, the Recursive Largest First (RLF) coloring algorithm. Adding onto RLF, an assortment of existing coloring methods are introduced and their execution on a wide scope of test data is contrasted with that of the RLF algorithm. Additionally portrayed is a methodology for producing arbitrary graphs with known chromatic number. The presence of such a technique, until now ailing in the test writing, gives a standard strategy to testing the exactness of graph coloring algorithms. Gend Lal et al. [4] describes an efficient algorithm for GCP colouring problem using less number of colours. The proposed scheme is applicable for all types of graphs. The algorithm divides the neighbours of a vertex into two categories N-type and V-type, and checks the colour filled in V-type neighbor before filling in the current vertex. Algorithm selects a colour from the list of colours, K every time from the beginning of the list colour so that it can make use less number of colours. They also compare their results with those obtained using genetic algorithms, Brown's algorithm and other heuristics algorithms. Michael Elkin et. al. [5] initiate the study of combinatorial algorithms for Distributed Graph Coloring problems. In a distributed setting a communication network is modeled by a graph $G = (V, E)$ of maximum degree Δ. The vertices of G host the processors, and communication is performed over the edges of G.

The goal of distributed vertex coloring is to color V with $(\Delta + 1)$ colors such that any two neighbors are colored with distinct colors. Currently, efficient algorithms for vertex coloring that require $O(\Delta + \log* n)$ time are based on the algebraic algorithm of Linial that employs set-systems. Evstigneev et. al. [6] shown that certain sequential coloring algorithm heuristics like largest-first (LF), smallest-last (SL), and saturation largest-first (SLF), as applied to some classes of graphs and to special cases of vertex coloring in distributed algorithms, produce an optimal or near-optimal coloring. Scott sallinen et al. [7] presents an event-based framework, and a novel, on the web, conveyed graph coloring algorithm. The execution for coloring static graphs, utilized as an execution pattern, is up to a request of size quicker than past outcomes and handles massive graphs with more than 257 billion edges. Our system bolsters dynamic graph coloring with execution everywhere scale superior to GraphLab's static examination. Our experience shows that online solutions are possible, and can be more effective than those dependent on snapshotting. Omari et. al [8] proposed two new algorithms in their paper work. They compared empirically in terms of used colors with some of the known heuristic graph coloring algorithm such as Largest Degree Ordering (LDO), First Fit (FF), saturated Degree Ordering (SDO), and incident degree ordering (IDO). Shilp et al [9] presents two parallel CREW (Concurrent Read Exclusive Write) PRAM algorithms for ideal coloring of general graphs on stream processing architectures, for example, the GPU. The algorithms are actualized utilizing OpenCL. The principal calculation introduces the strategies for processing vertex free sets on the GPU and after that appoints hues to them. The second calculation centers around the advancement of the vertex autonomous set calculation for edge-transitive charts by exploiting the structures of such diagrams and after that appoints coloring to every one of the standardized independent sets. In Gopalakrishnan et al [10], the paper centers around structuring three new transformative administrators utilizing Tabu seeking which are required to counterbalance the issues in the current understood strategies in minimal search space and generations, other than augmenting the level of fruitful goes through adequately conveying promising qualities for accomplishing quick stochastic combination with littler populace measure N. In the main technique, Single Parent Conflict Gene Extended Crossover and Conflict Gene Mutation with Advanced Local Guided Search administrators are planned and utilized. These administrators have been prepared with Conflict Gene Removal limitations in the second strategy to limit the pursuit space and to expand the level of effective keeps running of the hereditary calculation. Multipoint Single Parent Conflict Gene Crossover and Multipoint Conflict Gene Mutation with Advanced Local Guided Search administrators are utilized alongside a Conflict Gene Removal limitation in the third strategy. In [11], graph colouring issue emerges in various networking applications. It illuminates it in a completely decentralized manner (i.e., with no message passing). It proposes a novel calculation that is consequently receptive to topology changes, and this paper demonstrates that it converges to an appropriate colouring in O(N log N) time with high probability for generic graphs, when the quantity of accessible colours is more noteworthy than Δ, the maximum degree of the chart, and in O(log N) time if Δ = O(1). Mehmet Deveci et al [12] graph algorithms are trying to parallelize on many core architectures because of complex information conditions and sporadic memory access. This paper considers the very much examined issue of coloring the vertices of a graph. In numerous applications, it is essential to compute a coloring with few colors in near-linear time.

III. PARALLEL GRAPH COLORING ALGORITHM

Any algorithm that can color the set of vertices in parallel such that no two vertices are in parallel, then it is termed as Parallel Graph Coloring Algorithm. One of the well-known algorithm for graph coloring problems is Tabu search method [13].

A. *Tabu Search Method*

Tabu search, created by Fred W. Glover in 1986 and formalized in 1989, is a local scan strategy utilized for scientific improvement. Nearby inquiries take a potential answer for an issue and check its prompt neighbors (that is, arrangements that are comparable aside from a couple of minor subtleties) in the desire for finding an enhanced arrangement. Nearby hunt strategies tend to wind up stuck in imperfect locales or on levels where numerous arrangements are similarly fitted. Tabu pursuit improves the execution of these procedures by utilizing memory structures that depict the visited arrangements or client gave sets of guidelines. In the event that a potential arrangement has been recently visited in a certain short-term period or on the off chance that it has damaged a standard, it is set apart as "unthinkable" with the goal that the calculation does not think about that plausibility more than once.

B. *Performance Study*

We started with the analytical study of parallel processing on Tabu Search Algorithm. Based on our hypothesis, we assumed that parallel processing would be much faster and cause the coloring to have a shorter duration. Pj2 library [14] was used for parallelizing the algorithms on certain class of graph. Parallel Java 2 (PJ2) is an API and middleware for parallel programming in 100% Java on multicore parallel computers, cluster parallel computers, hybrid multicore cluster parallel computers, and GPU accelerated parallel computers [14]. It also includes a lightweight map-reduce framework.

Parallel Processing

The Tabu search algorithm implemented with several factors into consideration such as number of nodes in test input, number of cores for each coloring process, whether the task file was for parallel or sequential graph coloring. Total time (ms) taken for the given graph is measured by the number of nodes and cores. Below are the screenshots which depict the obtained results for different node size. Then, we tabulated these results and tried to find relationships with various parameters aforementioned.

5-Node Graph

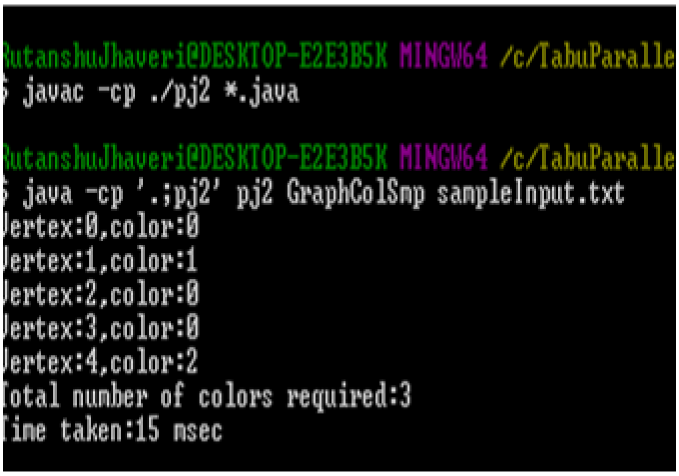


Fig. 1 Total time taken for graph coloring with node size 5 number of core used for implementation is 1

Core = 1, Node number = 5,
Time Taken = 15ms, Colors required = 3

Fig. 1 provides the result of the graph coloring using Tabu search method whose graph node size is 5. Total number of cores used during implementation is 1 and total number of color required is 3. The above scenario took 8ms for the completion of the coloring process. Fig. 2 provides the result of the same graph coloring process which is implemented with 3 cores and total number of color required is 3. It also took 8ms for the completion of the coloring process.

Cores = 2, Node number = 5,
Time Taken = 8ms, Colors required = 3

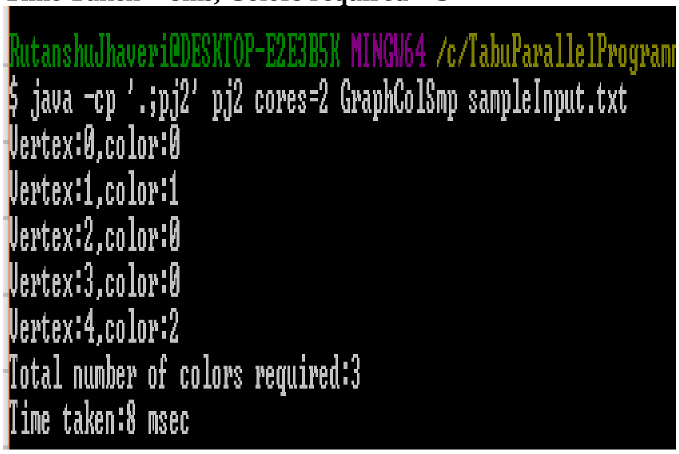


Fig. 2 Total time taken for graph coloring with node size 5 number of cores used for implementation is 3

Cores = 4, Node number = 5, Time Taken = 13ms, Colors required = 4

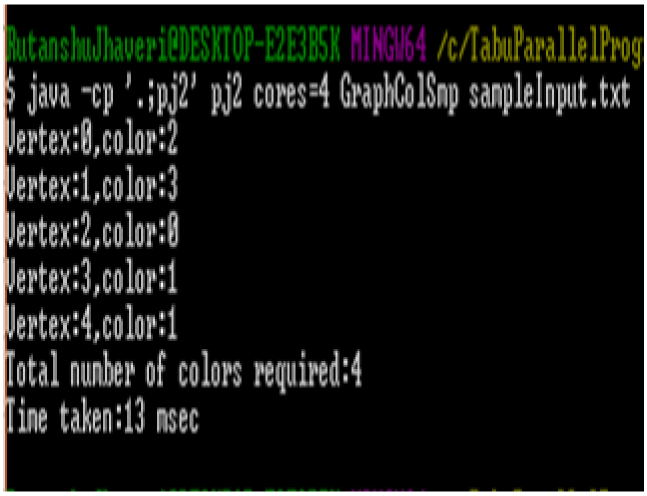


Fig. 3 Total time taken for graph coloring with node size 5 number of core used for implementation is 4

Fig. 3 provides the result of the graph coloring using Tabu search method whose graph node size is 5. Total number of cores used during implementation is 4 and total number of color required is 4. The above scenario took 13ms for the completion of the coloring process. We did not exceed number of cores more than four because then we were getting anomalous results with negative numbers as colors.

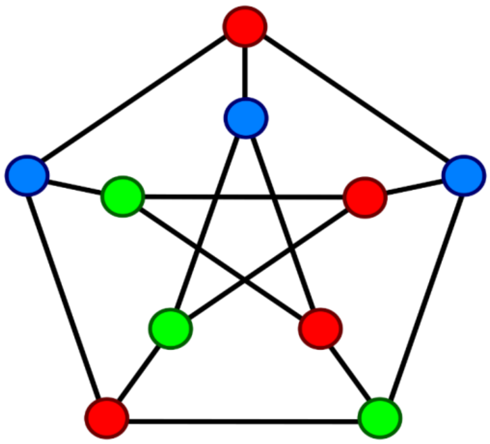


Figure 4 A 10-Node Graph

Fig.1.5 depicts a graph with 10 nodes. In the subsequent section, performance of graph coloring (with node size 10) through parallel processing is computed and the results are depicted as screenshots.

Core = 1, Node number = 10,
Time Taken = 103ms, Colors required = 3

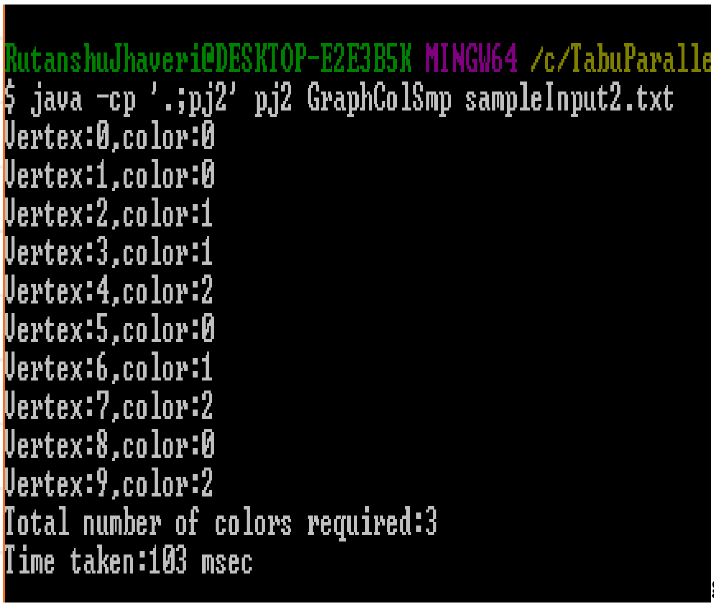


Fig. 5 Total time taken for graph coloring with node size 10 number of core used for implementation is 1

Fig. 5 shows the result of the graph coloring with node size 10 and is implemented with 1 core. Total number of color required is 3 and takes 103ms for the completion of the coloring process. Fig. 6 shows the result of the graph coloring with node size 10 and is implemented with 2 cores. Total number of color required is 3 and takes 105ms for the completion of the coloring process.

Cores = 2, Node number = 10,
Time Taken = 15ms, Colors required = 3

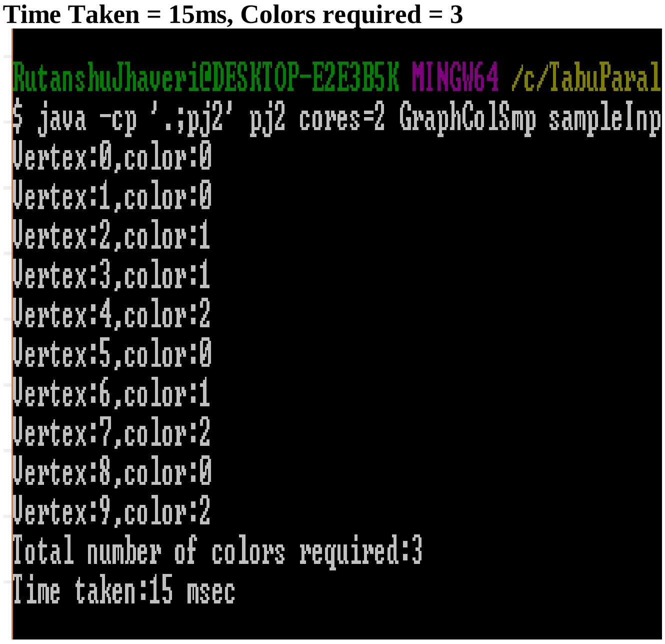


Fig. 6 Total time taken for graph coloring with node size 10 number of core used for implementation is 2

Fig. 7 shows the result of the graph coloring with node size 10 and is implemented with 1 core. Total number of color required is 4 and takes 9ms for the completion of the coloring process.

Cores = 4, Node number = 10,
Time Taken = 9ms, Colors required = 4

g. 7 Total time taken for graph coloring with node size 10 number of core used for implementation is 4

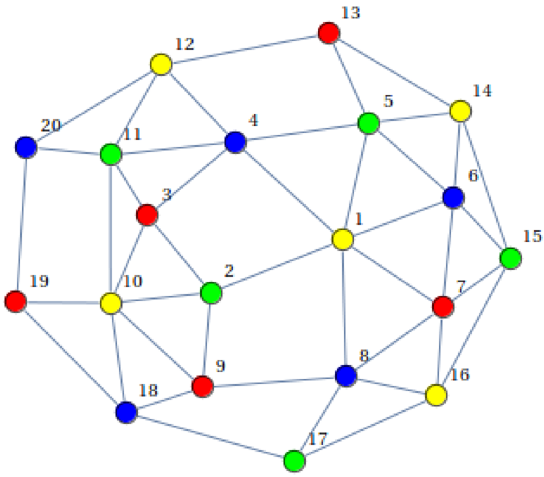


Figure 8 20-Node Graph

Fig.8 depicts a graph with 20 nodes. Now, performance of graph coloring (for node size 20) through parallel processing is computed and the results are depicted as screenshots.

Core = 1, Node number = 20,
Time Taken = 9ms, Colors required = 4

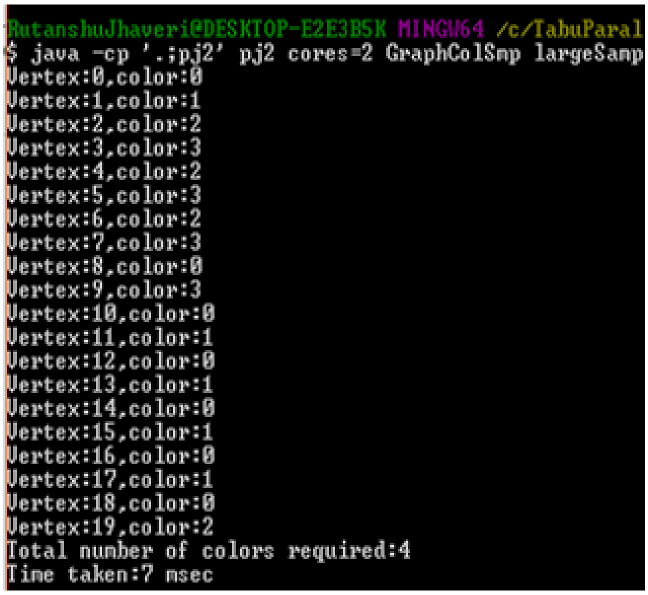
Fig. 10 shows the result of the graph coloring with node size 20 and is implemented with 1 core. Total number of color required is 4 and takes 9ms for the completion of the coloring process. Fig. 6 shows the result of the graph coloring with node size 10 and is implemented with 4 cores. Total number of color required is 4 and takes 7ms for the completion of the coloring process.

```
RutanshuJhaveri@DESKTOP-E2E3B5K MINGW64 /c/Ta
$ javac -cp ./pj2 *.java

RutanshuJhaveri@DESKTOP-E2E3B5K MINGW64 /c/Ta
$ java -cp '.;pj2' pj2 GraphColSmp largeSampl
Vertex:0,color:0
Vertex:1,color:1
Vertex:2,color:0
Vertex:3,color:1
Vertex:4,color:2
Vertex:5,color:1
Vertex:6,color:2
Vertex:7,color:1
Vertex:8,color:0
Vertex:9,color:2
Vertex:10,color:3
Vertex:11,color:0
Vertex:12,color:1
Vertex:13,color:0
Vertex:14,color:3
Vertex:15,color:0
Vertex:16,color:2
Vertex:17,color:1
Vertex:18,color:0
Vertex:19,color:1
Total number of colors required:4
Time taken:12 msec
```

Fig. 9 Total time taken for graph coloring with node size 20 number of core used for implementation is 1

Cores = 2, Node number = 20,
Time Taken = 7ms, Colors required = 4



```
RutanshuJhaveri@DESKTOP-E2E3B5K MINGW64 /c/TabuParal
$ java -cp '.;pj2' pj2 cores=2 GraphColSmp largeSamp
Vertex:0,color:0
Vertex:1,color:1
Vertex:2,color:2
Vertex:3,color:3
Vertex:4,color:2
Vertex:5,color:3
Vertex:6,color:2
Vertex:7,color:3
Vertex:8,color:0
Vertex:9,color:3
Vertex:10,color:0
Vertex:11,color:1
Vertex:12,color:0
Vertex:13,color:1
Vertex:14,color:0
Vertex:15,color:1
Vertex:16,color:0
Vertex:17,color:1
Vertex:18,color:0
Vertex:19,color:2
Total number of colors required:4
Time taken:7 msec
```

Fig. 10 Total time taken for graph coloring with node size 20, number of core used for implementation is 2

Cores = 4, Node number = 20,
Time Taken = 7ms, Colors required = 5

```
RutanshuJhaveri@DESKTOP-E2E3B5K MINGW64 /c/TabuPar
$ java -cp '.;pj2' pj2 cores=4 GraphColSmp largeSa
Vertex:0,color:2
Vertex:1,color:3
Vertex:2,color:1
Vertex:3,color:3
Vertex:4,color:4
Vertex:5,color:0
Vertex:6,color:1
Vertex:7,color:0
Vertex:8,color:1
Vertex:9,color:4
Vertex:10,color:0
Vertex:11,color:1
Vertex:12,color:0
Vertex:13,color:1
Vertex:14,color:3
Vertex:15,color:2
Vertex:16,color:1
Vertex:17,color:2
Vertex:18,color:1
Vertex:19,color:2
Total number of colors required:5
Time taken:7 msec
```

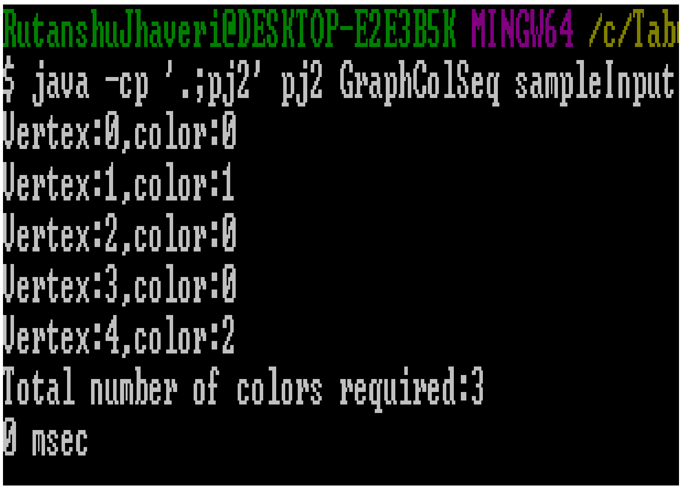
Fig. 11 Total time taken for graph coloring with node size 20, number of cores used for implementation is 4

Fig. 6 shows the result of the graph coloring with node size 20 and is implemented with 4 cores. Total number of color required is 5 and takes 7ms for the completion of the coloring process.

Tabu Search Algorithm – Sequential Processing

Below are the screenshots which shows the results of graph coloring by Tabu Search method through sequential processing. We were shocked to realize that most of the sequential processes gets completed in the time period of around 0ms-1ms.

Core = 1, Node number = 5,
Time Taken = 0ms, Colors required = 3



```
RutanshuJhaveri@DESKTOP-E2E3B5K MINGW64 /c/Tab
$ java -cp '.;pj2' pj2 GraphColSeq sampleInput
Vertex:0,color:0
Vertex:1,color:1
Vertex:2,color:0
Vertex:3,color:0
Vertex:4,color:2
Total number of colors required:3
0 msec
```

Fig. 12 Sequential process of the graph coloring process with 5 nodes and 1 core

Core = 1, Node number = 10,
Time Taken = 1ms, Colors required = 3

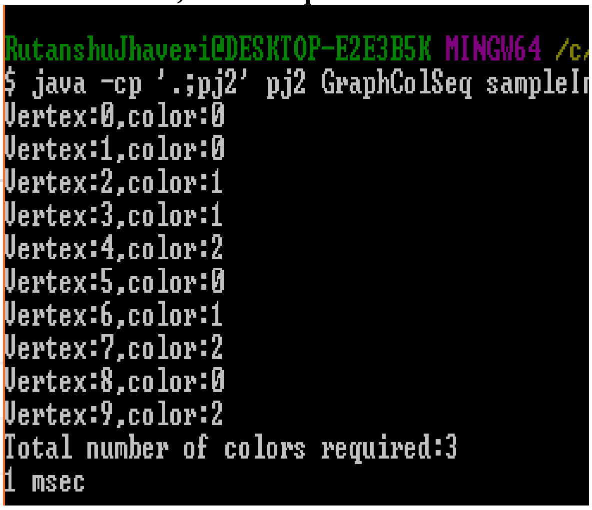


Fig. 13 Sequential process of the graph coloring process with 10 nodes and 1 core

Core = 1, Node number = 20,
Time Taken = 0ms, Colors required = 4

Fig. 14 Sequential process of the graph coloring process with 20 nodes and 1 core

Fig. 12, 13 and 14 shows the result of the graph coloring (sequential processing) with node size 5, 10 and 20 respectively and is implemented with 1 core.

Table 1 Consolidated results of Tabu parallel search

GraphColSmp			
Nodes In Graph	**Cores**	**Time/ms**	**Colors Required**
5	1	15	3
5	2	8	3
5	4	13	4
10	1	103	3
10	2	15	3
10	4	9	3
20	1	12	4
20	2	7	4
20	4	7	5

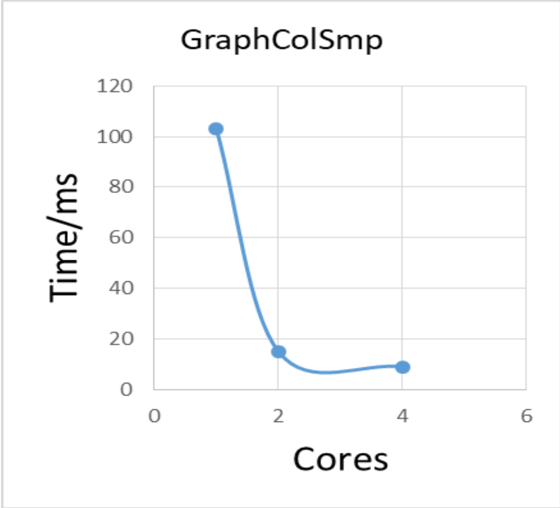


Fig. 15 Time v/s Cores graph

Fig.15 shows the graph of Time v/s Cores required for computation. It is clear that the time is inversely proportional to number of cores. So, more the cores for bigger graph reduces the computation time required. We have ignored the number of colors required out here since we only want to visualize the relationship between cores and time. The above bar graph considers all the parameters such as colors required, time and cores. The optimum balance between all these parameters is set when the number of core is two. In Table 2, we see that Sequential process takes almost negligible amount of time and this proves our hypothesis as false. We understood that the sequential processing may take more time because of overhead. In general, when people make sweeping statements about computer performance, there are far more variables at play here, and you can't really make that assumption. For example, inside your `for` loop, you are doing nothing more than `Math.Pow`, which the processor can perform very quickly. If this is an I/O intensive operation, requires each thread to wait for a long time, or even if it were a series of processor-intensive operations, you would get more out of parallel processing (assuming you have a multi-threaded processor). But as it is, the overhead of creating and synchronizing these threads is far greater than any advantage that parallelism might give you.

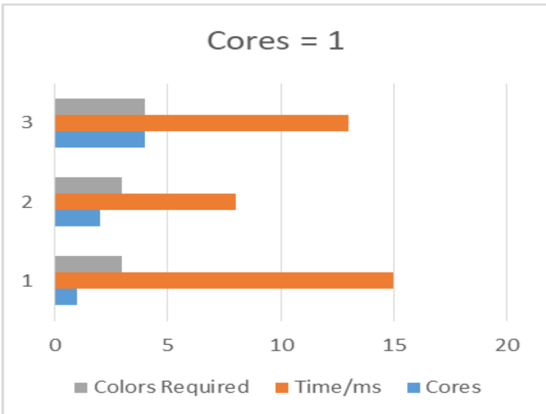


Fig. 16 Results for Tabu Sequential algorithm

Retrieval Number: B1840078219/19©BEIESP
DOI: 10.35940/ijrte.B1840.078219
Journal Website: www.ijrte.org

Published By:
Blue Eyes Intelligence Engineering
& Sciences Publication

Table 2 Consolidated results of graph coloring process during sequential processing

GraphColSmp			
Nodes In Graph	Cores	Time/ms	Colors Required
10	1	0	3
5	1	1	3
20	1	0	4

IV. CONCLUSION

In this paper, we discussed both serial and parallel implementation of Tabu search algorithm for a class of connected graphs. The performance of Tabu parallel search algorithm for graphs is comparatively better than the Tabu serial algorithm. In addition to that, the performance gets improved when the number of nodes keep increasing. On the other hand, Tabu serial search algorithm offer better performance when the size of the graph is relatively small. We can extend this study to all class of graphs with suitable restrictions.

REFERENCES

1. J.R.Allwright, R.Bordawekar, P.D.Coddington, K.Dincer and C.L.Martin, “A comparison of parallel graph coloring algorithms”, Northeast Parallel Architecture Center, Syracuse University, Tech. Rep., 1995
2. B.Chen, B.Chen, H.Liu and X.Zhang, “A Fast Parallel Genetic Algorithm for Graph Coloring Problem Based on CUDA”, in 2015 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC) IEEE, pp. 145-148, 2015.
3. F.T.Leighton, “A graph coloring algorithm for large scheduling problems”, Journal of research of the national bureau of standards, 84(6), pp.489-506, 1979.
4. G.L.Prajapati, A.Mittal and N.Bhardwaj, “An efficient colouring of graphs using less number of colours”, in IEEE Information and Communication Technologies (WICT), pp. 666-669, 2012.
5. L.Barenboim and M.Elkin, “Combinatorial algorithms for distributed graph coloring”, Distributed Computing, 27(2), pp.79-93, 2014.
6. V.A.Evstigneev, “Graph coloring in a class of parallel local algorithms”, Numerical Analysis and Applications, 4(3), pp.189, 2011.
7. S.Sallinen, K.Iwabuchi, S.Poudel, M.Gokhale, M.Ripeanu and R.Pearce, “Graph colouring as a challenge problem for dynamic graph processing on distributed systems” IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 347-358, 2016.
8. H.Al-Omari and K.E.Sabri, “New graph coloring algorithms”, American Journal of Mathematics and Statistics, 2(4), pp.739-741, 2006.
9. S.Sengupta, Parallel graph coloring algorithms on the GPU using OpenCL” IEEE International Conference on Computing for Sustainable Global Development (INDIACom), pp. 353-357, 2014
10. R.Marappan and G.Sethumadhavan, “Solution to Graph Coloring Using Genetic and Tabu Search Procedures”, Arabian Journal for Science and Engineering, 43(2), pp.525-542, 2018
11. Alessandro Checco and J.Doug Leith, “Fast, Responsive Decentralized Graph Coloring”, In IEEE/ACM Transactions on Networking, Vol. 25, No. 6, December 2017, Page(s): 3628 – 3640.
12. Mehmet Deveci, Erik G Boman, Karen D Devine, and Sivasankaran Rajamanickam, “Parallel Graph Coloring for Manycore Architectures” in IEEE International Parallel and Distributed Processing Symposium (IPDPS), 2016
13. Hertz and D. de Werra, “Using tabu search techniques for graph coloring”, computing, 39:345–351, 1987.
14. https://www.cs.rit.edu/~ark/pj2.shtml

AUTHORS PROFILE

Rutanshu Jhaveri completed his B.Tech Computer Science and Engineering at Vellore Institute of Technology, Vellore, India in the year 2019. His primary interest in the area of parallel and distributed computing, High performance computing and Graph coloring algorithms.

Dr. Narayanan Prasanth is an Associate Professor at School of Computer Science and Engineering, VIT Vellore India. He received his B.Tech IT from Pondicherry University, M.E CSE from Anna University and Ph.D CSE from M.S University Tirunelveli. His research interest includes Network Switch Scheduling, Switching architecture and SDN. He has published more than 20 papers in various conferences and journals. He is a life member of ISTE and member of CSTE.

Dr. Jayakumar Kaliappan received the B.E., degree in Computer Science and Engineering from M.K. University, India in 2002, M.E., degree in Computer Science and Engineering from Anna University, India in 2005 and Ph.D. degree in the field of Intrusion Detection Systems from Anna University, India in 2018. He is currently working as an Associate Professor at the School of Computing Science and Engineering, VIT University, Vellore, India. He has presented and published more than 20 papers in conferences and Journals. His current research interests include: Intrusion Detection Systems, Data mining and Machine Learning. He is a Life Member of Indian Society for Technical Education.

Dr. Navaz K pursued Bachelor of Technology and Master of Engineering from Anna University of Tamilnadu, India in year 2006 and 2009 respectively. He is also completed Ph.D from MSU Tirunelveli and currently working as Associate Professor in Department of Computer Science and Engineering, Annamacharya Institute of Technology and Sciences, Tirupathi, India. He is a member of IAENG & ICSES computer society. He has published more than seven research papers in reputed international journals including Thomson Reuters (SCI & Web of Science) and conferences. His main research work focuses on Computer Networks, Network Security, Cloud Security and Privacy, IoT and Computational Intelligence based education. He has 9 years of teaching experience and 5 years of Research Experience.