EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

A Solution to Graph Coloring Problem Using Genetic Algorithm

Karan Malhotra [1], Karan D Vasa [2], Neha Chaudhary[3,*], Ankit Vishnoi[4,] and Varun Sapra[5]

[1]Thirona, Nijmegen, 6525 EC, Netherlands
[2]Infosys, Pune 411057, Maharashtra, India
[3]Manipal University Jaipur 303007, India
[4]CSE Department, Graphic Era Deemed to be University, Dehradun, Uttarakhand, 248002, India
[5]School of computer science, University of Petroleum and Energy Studies, Dehradun 248007, India

Abstract

INTRODUCTION: The Graph Coloring Problem (GCP) involves coloring the vertices of a graph in such a way that no two adjacent vertices share the same color while using the minimum number of colors possible.
OBJECTIVES: The main objective of the study is While keeping the constraint that no two neighbouring vertices have the same colour, the goal is to reduce the number of colours needed to colour a graph's vertices. It further investigate how various techniques impact the execution time as the number of nodes in the graph increases.
METHODS: In this paper, we propose a novel method of implementing a Genetic Algorithm (GA) to address the GCP.
RESULTS: When the solution is implemented on a highly specified Google Cloud instance, we likewise see a significant increase in performance. The parallel execution on Google Cloud shows significantly faster execution times than both the serial implementation and the parallel execution on a local workstation. This exemplifies the benefits of cloud computing for computational heavy jobs like GCP.
CONCLUSION: This study illustrates that a promising solution to the Graph Coloring Problem is provided by Genetic Algorithms. Although the GA-based approach does not provide an optimal result, it frequently produces excellent approximations in a reasonable length of time for a variety of real-world situations.

Keywords: Genetic Algorithm, serial execution, parallel execution, graph colouring

Received on 19 December 2023, accepted on 09 March 2024, published on 15 March 2024

doi: 10.4108/eetsis.5437

*Corresponding author. Email: chaudhary.neha@jaipur.manipal.edu

1. Introduction

The Graph Coloring Problem (GCP) stands as a fundamental challenge in combinatorial optimization, with applications ranging from scheduling to network design. The Graph Coloring Problem (GCP) involves coloring the vertices of a graph in such a way that no two adjacent vertices share the same color while using the minimum number of colors possible. The GCP is known to be NP-hard, making it a computationally challenging task for large-scale graphs. As there is no known polynomial-time algorithm to solve it exactly. However, there are many heuristic algorithms that can find approximate solutions to the graph coloring problem. To address this complex optimization problem, researchers have explored various heuristic and metaheuristic approaches. One such promising technique is the Genetic Algorithm (GA), inspired by the process of natural selection and evolution. The ability of Genetic algorithms to handle vast search spaces and generating optimal solutions for the problems makes it a best fit for solving combinatorial problems like GCP.
Graph coloring has many applications, including:

• Scheduling: Graph coloring can be used to schedule tasks so that no two tasks that conflict with each other are scheduled at the same time. For example, graph coloring can be used to schedule classes at a school so that no two students are taking classes that meet at the same time.
• Register allocation: Graph coloring can be used to allocate registers in a compiler so that no two variables that are used at the same time are assigned to the same register.
• Channel assignment: Graph coloring can be used to assign channels in a wireless network so that no two nearby channels are used at the same time.

In this study, we suggest a novel use of GA for solving the Graph Colouring problem. The performance of the proposed method is evaluated by rigorously testing the method on different graph sizes. The method has been implemented on three different implementation settings - serial execution, high-specified parallel execution on Google Cloud so that the behaviour of the method can be analysed with the increase in the graph size as well as the effects of cloud computing resources and parallel processing on the overall execution time.

This work includes a thorough analysis of the obtained data, focusing on performance measures such as execution time and scalability. In particular, we investigate the variation of the execution time for each implementation parameter as the number of nodes in the graph increases. Comparing the three implementation schemes - serial, parallel, and parallel on Google Cloud provides us a deep insight regarding the efficiency achieved using parallelism and cloud computing resources.

2. Literature Review

In this paper, Dey, Arindam et al. present a genetic algorithm (GA) as a solution approach for the graph coloring problem (GCP). The authors propose a novel GA tailored to address the GCP efficiently [1]. Through rigorous experimental evaluations, the proposed GA demonstrates competitive performance in terms of solution quality and execution time. The paper contributes significantly to the field by introducing a GA-based technique for the GCP and provides valuable insights into the algorithm's effectiveness in handling large-scale graph instances.

S. Balakrishnan et al. propose a multi-objective genetic algorithm (MOGA) to address the graph coloring problem. The authors recognize that the GCP involves multiple conflicting objectives, including minimizing the number of colors and optimizing other relevant graph properties [2]. The MOGA optimizes these objectives simultaneously, providing a set of solutions representing different trade-offs between them. Experimental evaluations on various graph instances demonstrate that the MOGA approach produces a diverse set of high-quality solutions. The paper demonstrates the use of multi-objective optimization in case of GCP and showcase the significant benefits of multiple objectives in GA-based solutions.

Using a Reduced Quantum Genetic Algorithm, Ardelean et al. proposed a novel GA solution for the problem of graph colouring [3]. In their approach they propose a unique technique to enhance search space exploration by maintaining a consistent population size throughout the process. When compared to typical GAs, the fixed-size pool technique improves both convergence speed and solution quality. The performance of the graph is evaluated by experimental assessments on benchmark graphs by computing time and solution quality.

In this work, idi Mohamed Douiri et al. proposed a hybrid approach for solving the graph colouring problem (GCP) [4]. To represent graph colours, the authors suggested a hybrid approach that makes it possible to explore the solution space effectively. The performance of the suggested method in generating high-quality solutions was evaluated using advanced local guided search, with low computational overhead. The paper significantly advances the area by introducing a novel encoding method and showing the potential applications of real-coded GAs for the GCP.

A novel crossover operator based on complementing solutions is included in Marappan et al.'s [5] improved genetic algorithm for the GCP. The proposed operator promotes search space exploration by enabling the exchange of complementary genetic information among solutions. The novel crossover operator significantly increases solution diversity and convergence speed, as the authors' extensive testing on a variety of graph cases show. By comparing the algorithm's performance against traditional GAs, the benefits it offers in terms of execution time and solution quality become clear.

X. Li et al. propose a hybrid approach that combines a variable neighborhood search with a genetic algorithm to address the GCP [6]. The hybrid method utilizes VNS as a local search strategy to improve solutions generated by the GA. By incorporating the VNS within the GA's evolutionary framework, the algorithm effectively explores the solution space and refines solutions locally. Experimental evaluations on graphs demonstrate the hybrid approach's superior performance compared to stand-alone GAs and VNS methods. This paper offers valuable contributions by showcasing the synergy between VNS and GA, highlighting the potential of hybrid approaches for the GCP, and achieving competitive results in graph coloring tasks.

Hamed Kazemi et al. present a parallel hybrid genetic algorithm (GA) designed to tackle the graph coloring problem (GCP) in a distributed computing environment [7]. The authors combine the benefits of parallel processing and the GA's evolutionary mechanisms to improve solution efficiency. By distributing the GA's population across multiple computational nodes, the parallel hybrid approach achieves faster convergence and reduces the execution time for large-scale graph instances. Experimental evaluations demonstrate the effectiveness of the parallel hybrid GA in producing high-quality solutions

in comparison to traditional serial GAs. This paper contributes valuable insights into harnessing the power of parallelism for combinatorial optimization problems like the GCP.

Feng Liu et al. propose a dynamic multi-objective genetic algorithm for the rotating seru production problem [8]. The method incorporates a hybrid crossover operator that combines different crossover methods to balance exploration and exploitation in the solution space. The algorithm dynamically adapts its parameters during the evolutionary process to improve convergence and enhance solution diversity. The paper contributes to the advancement of multi-objective optimization techniques and highlights the benefits of dynamically adjusting GA parameters.

Feng et al. evaluated different approaches of genetic algorithm and compared them on the basis of chromatic number generated by each approach. The authors strongly recommended the utilization of the right GA approach provide faster convergence with minimum population size. [9]. Experimental evaluations conducted on graphs demonstrate the efficacy of the GA operator in enhancing the algorithm's performance. By exploiting the inherent graph structure, the genetic algorithm produces competitive results compared to conventional GAs. This paper contributes valuable insights into the problem-specific operators for the GCP, showcasing the potential of tailoring genetic algorithms to address optimization problems.

Mohamed, et al. introduce a novel genetic algorithm-based approach to address the optimization problems. The authors propose an innovative Gaining Sharing Knowledge scheme to solve optimization problems more efficiently [10]. The algorithm utilizes unique genetic operators tailored to the specific encoding, promoting effective exploration of the search space. Experimental evaluations on optimization demonstrate that the proposed approach achieves competitive results in terms of both solution quality and computation time. This paper contributes to the field by presenting a novel genetic algorithm GSK for the optimization.

R. Marappan et al. discussed the robust method to solve the NP Hard problem of graph coloring. This study proposed a method to implement the single parent and conflict gene mutation [11]. further it compared the time taken for the conversion with the existing methods and finds out the significant improvements.

Et al. discussed the results achieved after the implementation of Evolutionary algorithms to solve the graph coloring problem. This study compared the traditional GA's with the proposed method considering the parameters like edge density, topology and size [12]. The results conclude that the EA provides better stats while comparing it with the traditional algorithms.

Eiben et al. give a new method to solve the GCP using parallel approach which involves HPGA [13]. This research discussed the proposed method derived from VOA. They used DIMACS website to compare the proposed method and find out, it is better than the traditional approaches.

In 2016 R. Marappan et al. again discussed the single conflict gene and proposed a new procedure to solve the problem [14]. It again provides significantly improved results when compared with the las research and the modern GA.

Kong, Y et al. proposed RGCP method to solve the GCP which uses the method to provide real life solution for the problem. They also proposed the application that can be implemented to develop the application for timetable management [15].

Brighen et al. explains the use of distributed approach to solve the GCP for the larger graphs derived from VGCP [16]. They proposed a new algorithm "DistG". Among all the supersets it starts coloring from second superset, and provides some promising results.

3. Methodology

In order to assign frequencies to different nodes, we will utilize a Genetic Algorithm (GA). GA is a type of metaheuristic, that draws inspiration from natural selection and belongs to the broader category of evolutionary algorithms (EA). By employing bio-inspired operators like mutation, crossover, and selection, genetic algorithms are widely employed to produce excellent solutions for optimization and search problems.

Cross Over in Genetic Algorithm:

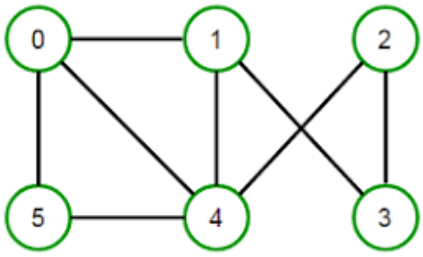

Figure 1: Sample Graph

In the context of the graph coloring problem and the given sequence, it appears that a genetic algorithm (GA) is being used to find a valid and optimized coloring for the graph. The goal is to assign colors to nodes (vertices) in such a way that no two adjacent nodes have the same color while minimizing the total number of colors used.

The given sequence represents different color assignments for the nodes of the graph. Each number represents a color assigned to an individual node. The sequence is structured as follows (Flow chart of the prosses given in figure 2):

- The graph has 6 nodes labeled as 0, 1, 2, 3, 4, and 5.
- The first set of numbers (e.g., 0-1-2-3-4-5) represents a specific coloring assignment for the graph, where node 0 is assigned color 0, node 1 is assigned color 1, node 2 is assigned color 2, and so on.

Two examples (E.g. -) of different valid colorings are given:

1. Example 1:
   • Nodes: 0-1-2-3-4-5
   • Colors: 1 – 2 – 1 – 3 – 3 – 4
2. Example 2:
   • Nodes: 0-1-2-3-4-5
   • Colors: 2 – 1 – 1 – 4 – 3 – 2

The GA proceeds with a crossover operation on these examples to create new color assignments. The crossover combines the color assignments of two-parent colorings to generate new potential solutions.

After the crossover, two new colorings are generated:

1. New Coloring 1:
   • Colors: 1 – 1 – 1 – 3 – 3 – 4
   • Result: Rejected, meaning this new coloring is invalid according to the graph coloring problem since nodes 1 and 2 are assigned the same color (1) and are adjacent.
2. New Coloring 2:
   • Colors: 1 – 2 – 1 – 4 – 3 – 4
   • Result: Correct, this new coloring is valid according to the graph coloring problem, and no adjacent nodes have the same color.
3. New Coloring 3:
   • Colors: 2 – 1 – 1 – 3 – 3 – 4
   • Result: Correct, this new coloring is also valid, and adjacent nodes have different colors.

The GA aims to find the best coloring that satisfies the constraints of the graph coloring problem while minimizing the number of colors used. The examples given illustrate how the GA iteratively explores different color assignments to eventually converge to a suitable and optimized solution.

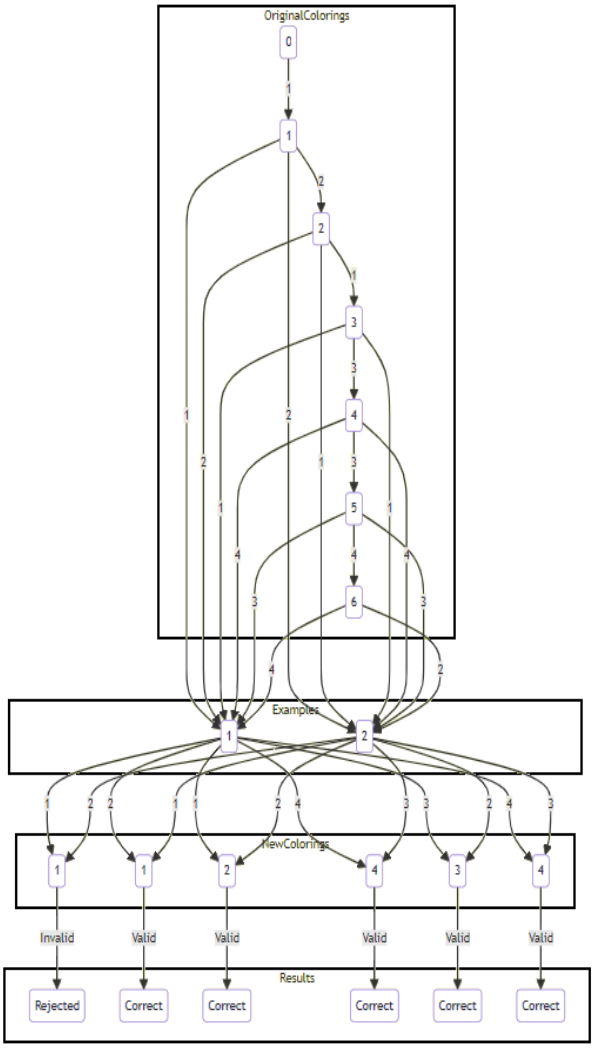

Fig. 2: Flow Chart of the GA for GCP with above Example

4. Results

The initial population is created through random generation, and the fitness of each individual is assessed by considering the conflicts that occur when adjacent vertices are assigned the same color. The crossover operation merges the genetic information of two-parent individuals to produce new offspring solutions, while the mutation operation introduces slight random alterations to preserve diversity within the population. Throughout the evolution process, the genetic algorithm aims to converge toward a feasible and efficient coloring of the graph, using as few colors as possible. The termination criteria can be defined either as a maximum number of generations or upon reaching a satisfactory solution.

Initial experiment was performed on a single processor/core by taking different nodes from 4 to 15 (all number of nodes values are taken randomly) and calculated the execution time. For each number of nodes,

an average value is calculated for 3 different iterations of execution.

Table 1: Serial execution of the proposed method for different numbers of nodes

No of Nodes	Serial Execution time
4	0.541
4	0.543
4	0.495
Average	0.53
5	6.456
5	3.184
5	4.286
Average	4.642
6	9.127
6	8.033
6	6.326
Average	7.82
10	10.763
10	30.34
10	5.022
Average	15.375
15	236.394
15	167.615
15	195.259
Average	199.756

The table 1 represent the results of serial execution (on a single processor/core), and figure 3 for different numbers of nodes (or data points). Each row represents a separate execution, and the "Execution time" column shows the time taken to complete the task for that specific number of nodes. The "Average" value at the bottom of each table represents the average execution time across all the executions for that particular number of nodes.

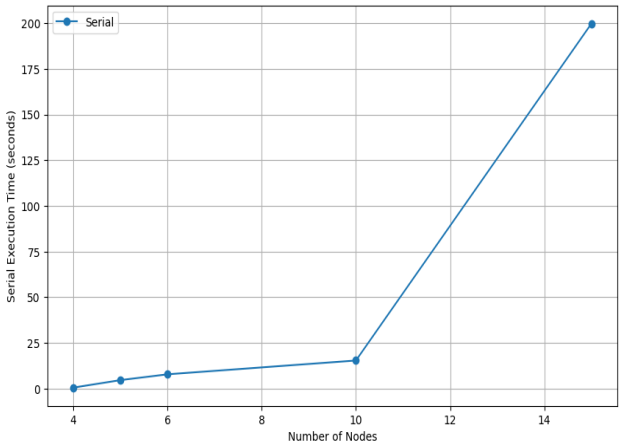

Fig. 3. Serial Execution Time vs Number of Cities

Later the experiment was performed on a parallel execution on multiple processors/cores by taking different nodes from 4 to 15 (all number of nodes values are taken randomly) and calculated the execution time. For each number of nodes, an average value is calculated for 3 different iterations of execution.

Table 2: Parallel execution of the proposed method for different numbers of nodes

No of Nodes	Parallel Execution time
4	0.222
4	0.522
4	0.236
Average	0.326
5	3.953
5	1.876
5	2.459
Average	2.76
6	3.452
6	5.918
6	9.459
Average	6.27
10	5.545
10	9.509
10	12.391
Average	9.14
15	100.209
15	92.381
15	122.391

Average	104.99

The table 2 represent the results of parallel execution on multiple processors/cores, and figure 4 for different numbers of nodes (or data points). Each row represents a separate execution, and the "Execution time" column shows the time taken to complete the task for that specific number of nodes.

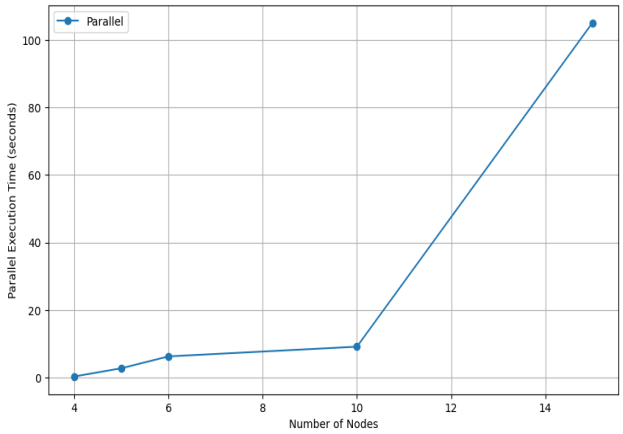

Fig. 4. Parallel Execution Time vs Number of Cities

Later the experiment was performed on the Google Cloud a cloud computing platform by taking different nodes from 4 to 15 (all number of nodes values are taken randomly) and calculated the execution time. For each number of nodes, an average value is calculated for 3 different iterations of execution.

Table 3: Google Cloud execution of the proposed method for different numbers of nodes

No of Nodes	Execution time on Google Cloud
4	0.247
4	0.294
4	0.213
Average	0.251
5	1.546
5	0.854
5	0.746
Average	1.04
6	2.114
6	4.122
6	3.711
Average	3.31
10	1.5
10	5.487
10	3.389
Average	3.45
15	65.713
15	53.491
15	61.301
Average	60.168

The table 3 represent the results Google Cloud on a cloud computing platform, and figure 5 for different numbers of nodes (or data points). Each row represents a separate execution, and the "Execution time" column shows the time taken to complete the task for that specific number of nodes.

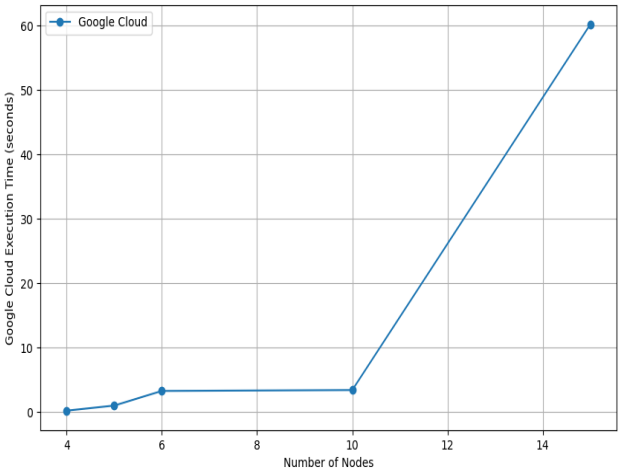

Fig. 5. Google Cloud Execution Time vs Number of Cities

The result of different executions shown above explain the time taken by the processes with different nodes. Now when the comparison of these different processes, serial execution (on a single processor/core), parallel execution (on multiple processors/cores), and Google Cloud (a cloud computing platform) was done it was found that the Google Cloud platform provides promising results. To perform the comparison the Average value of each execution is taken into consideration.

Table 4: Average Execution Time on Serial execution, Parallel execution and Google Cloud execution

No of Nodes	Average Serial Execution time	Average Parallel Execution time	Average Execution time on Google Cloud
4	0.53	0.326	0.251
5	4.642	2.76	1.04
6	7.82	6.27	3.31
10	15.375	9.14	3.45
15	199.756	104.99	60.168

Table 3 represents the average execution time of the process with different number on nodes asper serial execution, parallel execution and Google Cloud execution. Where as the figure 6 represents the comparison of these results.

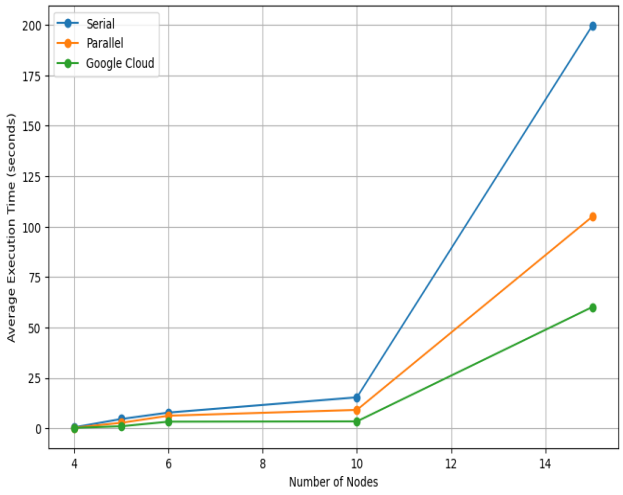

Fig 6: Average Execution Time Comparison

The figure 6 represents a comparison graph between serial execution, parallel execution, and parallel execution on the Google Cloud platform. It shows how the execution times vary across different configurations and numbers of nodes.

From the above comparison it is concluded that the Google Cloud platform provides most promising results when compared with Serial and Parallel execution on the machine. After receiving promising results from the Google Cloud execution, the experiments were extended to observe the execution time of the proposed method on Google Cloud with number of nodes between 15 to 75 (all number of nodes values are taken randomly).

Table 4: Exemplary Results Obtained on Google Cloud

No. of Nodes	Execution time on Google Cloud
15	3.513
20	7.559
30	20.971
50	34.497
75	138.35

The table 4 represents exemplary results obtained on Google Cloud for various numbers of nodes. The "Execution time" column shows the time taken to complete the task for each specific number of nodes.

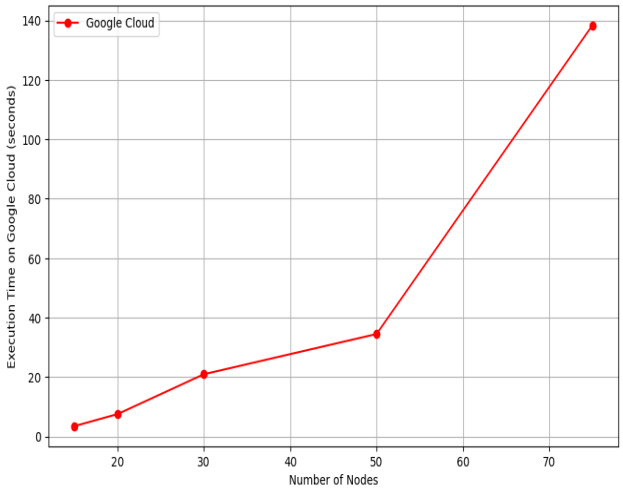

Fig. 7: Exemplary Results Obtained on Google Cloud

Figure 7 represents the number of nodes at the X axis and execution time in seconds of Y axis. After analysing the above figure, it is worth mentioning that Google Cloud is providing better result even while increasing the number of nodes to 75 as the execution time not going beyond 140 seconds.

5. Conclusion

This paper illustrates that a promising solution to the Graph Coloring Problem is provided by Genetic Algorithms. Although the GA-based approach does not provide an optimal result, it frequently produces excellent approximations in a reasonable length of time for a variety of real-world situations. In order to demonstrate the GA's competitiveness and promise as an efficient tool for addressing the Graph Coloring Problem in a variety of application domains, its computational efficiency and solution quality are examined. Based on figure 2 to 6, one can infer that the performance of the solution differs

depending on its implementation, whether serially, in parallel, or on the high-specification Google Cloud platform. It is evident that the serial implementation shows a significant increase in execution times as the number of nodes increases. In comparison, the rise is less steep for parallel execution and even less so when using the high-specification Google Cloud system. This solution can be implemented in various real-life scenarios like timetable management using the proposed method. For the future, it would be interesting to analyse the other possible areas where the above method can be applicable.

References

[1] Dey, Arindam et al. 'A Genetic Algorithm for Total Graph Coloring', Journal of Intelligent & Fuzzy Systems, vol. 37, no. 6, pp. 7831-7838, 2019.

[2] S. Balakrishnan, Tamilarasi Suresh, Raja Marappan. (2021) A New Multi-Objective Evolutionary Approach to Graph Coloring and Channel Allocation Problems. Journal of Applied Mathematics and Computation, 5(4), 252-263.

[3] Ardelean, Sebastian Mihai, and Mihai Udrescu. "Graph coloring using the reduced quantum genetic algorithm." PeerJ. Computer science vol. 8, e836, Jan. 2022, doi:10.7717/peerj-cs.836.

[4] idi Mohamed Douiri, Souad Elbernoussi, "Solving the graph coloring problem via hybrid genetic algorithms", Journal of King Saud University - Engineering Sciences, Volume 27, Issue 1, 2015, Pages 114-118, ISSN 1018-3639, https://doi.org/10.1016/j.jksues.2013.04.001.

[5] Marappan, R., Sethumadhavan, G. Solution to Graph Coloring Using Genetic and Tabu Search Procedures. Arab J Sci Eng 43, 525–542 (2018). https://doi.org/10.1007/s13369-017-2686-9.

[6] X. Li, L. Gao, Q. Pan, L. Wan and K. -M. Chao, "An Effective Hybrid Genetic Algorithm and Variable Neighborhood Search for Integrated Process Planning and Scheduling in a Packaging Machine Workshop," in IEEE Transactions on Systems, Man, and Cybernetics: Systems, vol. 49, no. 10, pp. 1933-1945, Oct. 2019, doi: 10.1109/TSMC.2018.2881686.

[7] Hamed Kazemi, Mohammad MahdaviMazdeh, Mohammad Rostami & Mahdi Heydari (2021) The integrated production-distribution scheduling in parallel machine environment by using improved genetic algorithms, Journal of Industrial and Production Engineering, 38:3, 157-170, DOI: 10.1080/21681015.2020.1848930.

[8] Feng Liu, Kan Fang, Jiafu Tang, Yong Yin, "Solving the rotating seru production problem with dynamic multi-objective evolutionary algorithms", Journal of Management Science and Engineering, Volume 7, Issue 1, 2022, Pages 48-66, ISSN 2096-2320, https://doi.org/10.1016/j.jmse.2021.05.004.

[9] R. Marappan and G. Sethumadhavan, "Complexity Analysis and Stochastic Convergence of Some Well-known Evolutionary Operators for Solving Graph Coloring Problem," Mathematics, vol. 8, no. 3, p. 303, Feb. 2020, doi: 10.3390/math8030303.

[10] Mohamed, A.W., Hadi, A.A. & Mohamed, A.K. Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm. Int. J. Mach. Learn. & Cyber. 11, 1501–1529 (2020). https://doi.org/10.1007/s13042-019-01053-x.

[11] R. Marappan and G. Sethumadhavan, "A New Genetic Algorithm for Graph Coloring," *2013 Fifth International Conference on Computational Intelligence, Modelling and Simulation*, Seoul, Korea (South), 2013, pp. 49-54, doi: 10.1109/CIMSim.2013.17.

[12] Eiben, A., van der Hauw, J. & van Hemert, J. Graph Coloring with Adaptive Evolutionary Algorithms. Journal of Heuristics 4, 25–46 (1998). https://doi.org/10.1023/A:1009638304510.

[13] Eiben, A., van der Hauw, J. & van Hemert, J. Graph Coloring with Adaptive Evolutionary Algorithms. Journal of Heuristics 4, 25–46 (1998). https://doi.org/10.1023/A:1009638304510.

[14] R. Marappan and G. Sethumadhavan, "Solution to graph coloring problem using divide and conquer based genetic method," *2016 International Conference on Information Communication and Embedded Systems (ICICES)*, Chennai, India, 2016, pp. 1-5, doi: 10.1109/ICICES.2016.7518911.

[15] Kong, Y., Wang, F., Lim, A., Guo, S. (2003). A New Hybrid Genetic Algorithm for the Robust Graph Coloring Problem. In: Gedeon, T. (D., Fung, L.C.C. (eds) AI 2003: Advances in Artificial Intelligence. AI 2003. Lecture Notes in Computer Science(), vol 2903. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-24581-0_11.

[16] Brighen, A., Slimani, H., Rezgui, A. et al. A new distributed graph coloring algorithm for large graphs. Cluster Comput (2023). https://doi.org/10.1007/s10586-023-03988-x.