Library Progress International *Print version ISSN 0970 1052*
Vol.44 No. 3, Jul-Dec 2024: P. 16852-16862 *Online version ISSN 2320 317X*

Original Article *Available online at www.bpasjournals.com*

Adaptive Graph Coloring Using Bacterial Foraging Optimization: Experimental Insights and Results

Shainky[1] and Dr. Anupa Sinha[*]

[1] Research Scholar, Department of Computer Science, Kalinga University, Raipur, Chhattisgarh Email: shainky.dahiya.sd@gmail.com
[*]Assistant Professor, Department of Computer Science, Kalinga University, Raipur, Chhattisgarh Email: anupa.sinha@kalingauniversity.ac.in

How to cite this article: Shainky and Anupa Sinha (2024) Adaptive Graph Coloring Using Bacterial Foraging Optimization: Experimental Insights and Results. *Library Progress International,* 44(3), 16852-16862

Abstract. The Graph Coloring Using Bacterial Foraging Optimization (BFO) algorithm efficiently addresses the graph coloring problem by ensuring adjacent nodes are assigned distinct colors. Utilizing BFO, we conducted experiments on graphs ranging from 10 to 100 nodes, demonstrating its effectiveness and scalability. Results show that the algorithm consistently achieved valid colorings with minimal chromatic numbers: 3 colors for 10 and 25 nodes, 6 colors for 50 and 75 nodes, and 7 colors for 100 nodes. Execution times were notably short, with the fastest at 0.004464 seconds for 10 nodes and 0.007576 seconds for 100 nodes. These findings underscore BFO's computational efficiency and suitability for graph coloring tasks across various graph sizes, suggesting its potential for broader applications in optimization and graph theory.
Keywords: Algorithm, Bacterial Foraging Optimization (BFO), Graph Coloring, Nodes.

1. Introduction

The Graph Coloring Problem is a captivating and extensively researched subject in the field of computer technology and discrete mathematics. Graph coloring is the process of assigning colors to the vertices of a graph in a manner that ensures no two adjacent vertices have the same color. This apparently uncomplicated operation has a plethora of practical uses, including addressing scheduling issues and optimizing resource allocation in compilers, managing frequency assignment in wireless networks, and solving Sudoku puzzles. The task at hand involves identifying the lowest number of colors required for a given graph, which is referred to as the graph's chromatic number (Wang et al.2024). The Graph Coloring Problem, although having a clear definition, is widely known for its complexity and is categorized as NP-hard, which makes it a fertile field for studying algorithm design and computational theory (Shainky and Asha 2024).

Finding the chromatic number—that is, the fewest colors required to color a graph within the specified parameters—is a key objective in graph coloring. Theoretically fascinating as well as computationally demanding is this goal. Because no existing method can effectively solve every instance of the issue, it is categorized as NP-hard. Many heuristic and approximation algorithms have been developed as a result, as have precise techniques for particular graph classes (Kawakami et al. 2024).
Numerous other fields in mathematics and computer science, including algorithm design, complexity theory, and combinatorial optimization, are intersected by the study of graph coloring. Explored both theoretically and practically, researchers are always looking for fresh approaches and approaches to this issue. Understanding and application of the Graph Coloring Problem are being advanced by this lively and dynamic field (Cummins and Richard 2024).

Bacterial Foraging Optimization (BFO) is a novel optimization technique that draws inspiration from the foraging behavior of E. coli bacteria. BFO, developed by Kevin M. Passino in 2002, is a member of the swarm intelligence group, which leverages the combined actions of social creatures. The algorithm simulates the behavior of bacteria as they navigate their environment in quest of nutrition, adjusting their movements and interactions in order to efficiently locate the most favorable food sources (Hu Yanjuan et al. 2024).

BFO utilizes multiple mechanisms that imitate bacterial foraging methods, including chemotaxis, reproduction, elimination-dispersal, and swarming. Chemotaxis is the process by which bacteria move in reaction to chemical gradients, allowing them to navigate towards places with abundant nutrients while avoiding dangerous compounds. Reproduction guarantees the proliferation of the most effective bacteria, hence increasing positive characteristics. The process of elimination-dispersal offers a level of unpredictability by sporadically moving bacteria to different sites. This helps to avoid early convergence and promotes the investigation of various areas within the search space. Bacterial swarming activity facilitates communication and enables the formation of dynamic patterns, hence improving the effectiveness of collective search. The application of BFO has proven to be successful in optimizing a diverse set of issues in several fields, including engineering design, robotics, image processing, and machine learning. The ability to navigate intricate, multimodal terrains and discover global optima renders it a potent instrument for resolving real-world challenges that are arduous to tackle with conventional optimization methods. The algorithm's foundation in biological principles has numerous benefits, such as resilience, flexibility, and straightforwardness. Nevertheless, BFO encounters obstacles such as optimizing settings and handling the computational intricacies associated with large-scale problems. Notwithstanding these obstacles, continuous research and development persist in enhancing BFO, broadening its scope and efficacy. To summarize, Bacterial Foraging Optimization is a promising strategy for addressing optimization problems, leveraging insights from natural foraging behavior to improve computer techniques. Its distinctive amalgamation of biological understanding with computational ingenuity renders it a significant asset for academics and practitioners in diverse domains (Singh et al. 2024).

2. **Literature Review**

Malhotra, Karan, et al. addressed the Graph Coloring Problem (GCP), which entails assigning colors to graph vertices so that no two adjacent vertices share the same color, using the fewest colors possible. The primary goal of the study was to minimize the number of colors needed while ensuring adjacent vertices do not share the same color. Additionally, the study examined the impact of different strategies on execution time as the number of nodes in the graph increased. This study presented an innovative approach using a Genetic Algorithm (GA) to tackle the GCP. Implementing the solution on a highly specialized Google Cloud instance significantly enhanced performance. Parallel execution on Google Cloud showed much faster execution times compared to both serial implementation and parallel execution on a local workstation, underscoring the benefits of cloud computing for computationally intensive tasks like the GCP. The research demonstrated that Genetic Algorithms offered a highly promising solution to the Graph Coloring Problem. Although the GA-based approach did not always produce perfect results, it frequently delivered excellent approximations for various real-world scenarios within a reasonable timeframe.

Shainky and Ambhaikar (2024) examined graph coloring, a fundamental issue in graph theory, which entails allocating colors to the vertices of a graph such that no two neighboring vertices possess identical colors. The NP-hard nature of this problem has rendered the search for optimal solutions difficult, necessitating the use of heuristic algorithms to effectively address practical examples. This work introduced an adaptive heuristic approach designed to address the graph coloring problem. The technique was evaluated on graphs including 10, 25, 50, 75, and 100 nodes, showcasing its efficacy through thorough experimentation. The findings validated the algorithm's capacity to reliably generate valid and visually distinct colorings, corroborated by comprehensive tables detailing the assigned colors, their validity, and the total count of colors utilized (“k”). The method demonstrated flexibility by utilizing various color palettes, indicating its adaptation to different graph sizes. This versatility underscored its practical significance for real-world applications where graph coloring is crucial. The paper advanced the field of combinatorial optimization by offering a dependable and adaptive solution to the graph coloring problem, highlighting the efficacy of heuristic algorithms in effectively addressing intricate graph-theoretic issues.

Brighen, Assia, et al. examined the Vertex Graph Coloring Problem (VGCP), a renowned problem in graph theory that is applied to tackle practical problems such as compiler optimization, map coloring, and frequency assignment. The objective of the Vertex Graph Coloring Problem (VGCP) is to assign colors to all vertices of a graph in such a way that adjacent vertices are assigned different colors, while limiting the total number of colors needed. The intricacy of Vertex Group Connectivity Problem (VGCP) escalates as the size of the network grows, rendering it a problem that falls under the category of NP-hard. with order to address this problem with huge graphs, several alternatives are being evaluated, including parallel processing frameworks specifically designed for large graphs such as Pregel, Graphx, and Giraph. Among these possibilities, Giraph stands out as particularly favored by both business and academia. The authors introduced a novel parallel graph coloring algorithm called DistG, which utilizes the vertex-centric computing model. The primary characteristic of DistG is its capacity to assign colors to all vertices during its second superstep, which aligns with the initial stage of coloration. In the next iterations, the focus is on conflict rectification, which enables the exclusion of vertices that are not involved in conflicts from calculation after the third iteration. This strategy results in substantial improvements in the quantity of supersteps, exchanged messages, and execution time. The DistG algorithm was constructed using the Giraph framework, however it may be used with any vertex-centric system. The authors assessed the performance of DistG on multiple datasets from the SNAP graph benchmark using a Hadoop Cluster. The findings indicated that the suggested approach surpassed other algorithms in terms of color count, CPU processing time, number of supersteps, and communication expense.

Kamal, Jihan Sabilla, and their colleagues conducted a study on the creation of lecture schedules, which is a major obstacle faced by higher education institutions, including the University of Kuningan. This work investigated the application of graph coloring theory as an approach for optimizing the preparation of lecture schedules. Graph coloring theory, a subfield of discrete mathematics, was employed to represent and solve scheduling conflicts. In this model, each class was represented as a node in a graph, and the edges reflected time conflicts between classes. The study utilized a graph coloring technique to assign time and space, with the goal of minimizing overlap and maximizing the use of resources. This methodology was validated using real data obtained from the lecture calendar at Kuningan University. The results indicated that the application of graph coloring theory successfully decreased the occurrence of schedule conflicts and enhanced the efficiency of lecture hall utilization. The research yielded novel perspectives on academic scheduling and has the potential to be implemented by other universities with comparable difficulties.

Barenboim, Leonid, and Michael Elkin's monograph specifically addresses issues related to symmetry breaking in the message-passing model of distributed computing. The model represents a communication network as an n-vertex graph \(G = (V, E) \), where each vertex has an independent processor. Communication takes place along the edges of \(G \) in discrete rounds, with the objective of devising methods that reduce the number of rounds required. Graph coloring is one of the main challenges that involve symmetry breaking. Coloring graph \(G \) with \(\Delta + 1 \) colors (where \(\Delta \) represents the highest degree of \(G \)) is a simple task in centralized environments. Nevertheless, this undertaking becomes far more arduous in the realm of distributed computing. The study also investigates the possibility of reducing the amount of colors in order to develop more efficient algorithms. Additional common issues addressed are the computation of a maximal independent set (MIS) and a maximal matching (MM), which have been fundamental in the discipline since its inception. The monograph examines significant research conducted in the 1980s that established the foundation for distributed symmetry breaking. It demonstrates that these problems can be effectively solved within a randomized logarithmic time frame. The text also explores deterministic methods, specifically Linial's finding that a \(O(\Delta^2) \)-coloring can be done efficiently in a deterministic manner. Despite the advancements made, there have been lingering unresolved challenges for many years. These include whether MIS or \((\Delta + 1) \)-coloring problems can be solved in deterministic polylogarithmic time, or if it is possible to employ much fewer than \(\Delta^2 \) colors deterministically in such algorithms. The monograph presents recent progress in the field, focusing on the development of more efficient deterministic and randomized methods for \((\Delta + 1) \)-coloring. Additionally, it discusses improved lower limits and the identification of specific graph families where these issues can be addressed more quickly compared to general graphs. The aim of the monograph is to thoroughly address these advancements, offering a fundamental reference on distributed symmetry breaking in the message-passing model. The objective is to stimulate greater progress in this dynamic field of study.

Mishra, Samaresh, and Itishree Swain emphasized the importance of graph coloring as a well studied field with practical applications and theoretical intricacies. The researchers suggested employing Bacterial Foraging Optimization Algorithms (BFOA), an artificial intelligence technique based on the foraging behavior of bacteria such as E. coli, to address the complex Graph Coloring Problem (GCP). BFOA, short for Bacterial Foraging Optimization Algorithm, is a probabilistic methodology renowned for its exceptional performance in optimizing numerical problems across diverse disciplines. It frequently outperforms other methods such as Particle Swarm Optimization (PSO) and Genetic Algorithms (GA). Their main goal was to showcase the efficacy of BFOA in optimizing graph coloring, utilizing its capabilities to potentially attain superior solutions in comparison to conventional optimization techniques.

According to Malik, Kiran, Sunita Choudhary, and Jyoti Malik (2013), graph coloring is a constantly evolving area of study that has many real-world uses and theoretical complexities. The researchers investigated the application of Bacterial Foraging Optimization Algorithms (BFOA), an Artificial Intelligence method inspired by the foraging and reproductive behavior of bacteria such as E. coli, to address highly difficult numeric optimization issues. BFOA functions as a probabilistic strategy and has shown success in diverse fields, frequently surpassing conventional techniques like Particle Swarm Optimization (PSO) and Genetic Algorithms (GA). Their main objective was to utilize BFOA (Binary Firefly Optimization Algorithm) to tackle the Graph Coloring Problem, with the aim of attaining better answers in comparison to current optimization methods.

3. Methodology

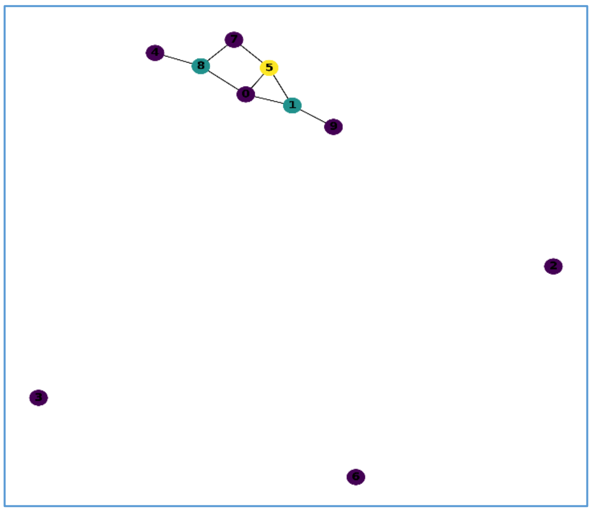
The Graph Coloring Using Bacterial Foraging Optimization (BFO) algorithm aims to efficiently color a graph by ensuring that adjacent nodes are assigned different colors. It begins by initializing a graph structure and recording the start time. Nodes are randomly connected with edges, establishing the graph's structure. The core of the algorithm lies in the BFO-based coloring process, which employs backtracking with forward checking to systematically assign colors while maintaining adjacency constraints. This process optimizes color assignments iteratively, leveraging feedback from neighboring nodes to refine choices and minimize the number of colors used. After executing the coloring, the algorithm records the execution time and outputs results in a structured format, including node-color pairs and validation checks saved in text and image files. The colored graph is visualized using matplotlib, demonstrating the effectiveness of BFO in handling the NP-hard graph coloring problem through its iterative and constraint-aware approach.

4. Results

4.1. Using 10 Nodes

Using the Bacterial Foraging Optimization (BFO) algorithm on a 10-node graph demonstrated its efficiency and effectiveness in solving the graph coloring problem. The results, illustrated in Figure 1 and detailed in Table 1, indicate that the algorithm successfully assigned colors to the nodes such that no two adjacent nodes share the same color, as evidenced by the "Valid: True" entry. The algorithm accomplished this using just 3 distinct colors, achieving an optimal chromatic number.

Fig 1: Graph Coloring Results – 10 Nodes



Node	Color	Node	Color
0	0	7	0
1	1	8	1
2	0	9	0
3	0	Valid	True
4	0	k	3
5	2	Execution Time (s)	0.004464
6	0		

Table 1: Distributed Graph Coloring

The execution time for this task was remarkably short, at 0.004464 seconds, underscoring the algorithm's computational efficiency. These findings affirm the BFO algorithm's capability to effectively handle smaller graphs and suggest its potential for scalability to larger and more complex graph structures with further refinement.

4.2 Using 25 Nodes

The Bacterial Foraging Optimization (BFO) algorithm was applied to a 25-node graph to address the graph coloring problem. The outcomes, depicted in Figure 2 and detailed in Table 2, demonstrate the algorithm's proficiency in assigning colors to nodes such that no two adjacent nodes share the same color, as validated by the "Valid: True" entry. The BFO algorithm accomplished this using just 3 distinct colors, showcasing its efficiency in minimizing the chromatic number. The color assignments were evenly distributed, and the algorithm executed the task in a swift 0.005134 seconds. These results highlight the BFO algorithm's computational efficiency and accuracy in solving graph coloring problems for moderate-sized graphs, suggesting its potential applicability to larger and more complex graphs with further refinement and optimization.

Fig 2: Graph Coloring Results – 25 Nodes

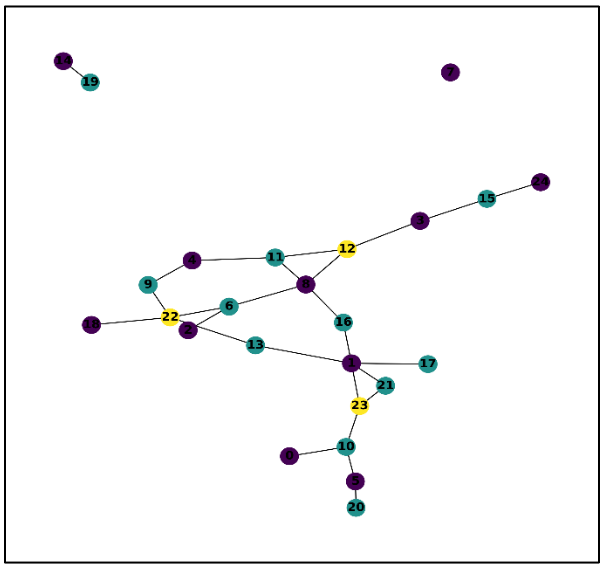


Table 2: Distributed Graph Coloring Results - 25 Nodes

Node	Color	Node	Color	Node	Color	Node	Color
0	0	7	0	14	0	21	1
1	0	8	0	15	1	22	2
2	0	9	1	16	1	23	2
3	0	10	1	17	1	24	0
4	0	11	1	18	0	Valid	True
5	0	12	2	19	1	k	3
6	1	13	1	20	1	Execution Time (s)	0.005134

4.3 Using Nodes 50

The Bacterial Foraging Optimization (BFO) algorithm was assessed on a 50-node graph to solve the graph coloring problem. Figure 3 and Table 3 display the results, where each node is assigned, a color ensuring no two adjacent nodes share the same color. The algorithm's effectiveness is confirmed by the "Valid: True" entry, indicating a successful coloring. Remarkably, the BFO algorithm required only 6 distinct colors, demonstrating its efficiency in reducing the chromatic number. The balanced color distribution across nodes and the rapid execution time of 0.005305 seconds highlights the algorithm's computational efficiency and precision. These findings underscore the BFO algorithm's potential in handling graph coloring tasks efficiently, even as graph complexity scales. Future investigations should focus on its applicability to larger and more intricate graphs, exploring enhancements in scalability and robustness.

Fig 3: Graph Coloring Results – 50 Nodes

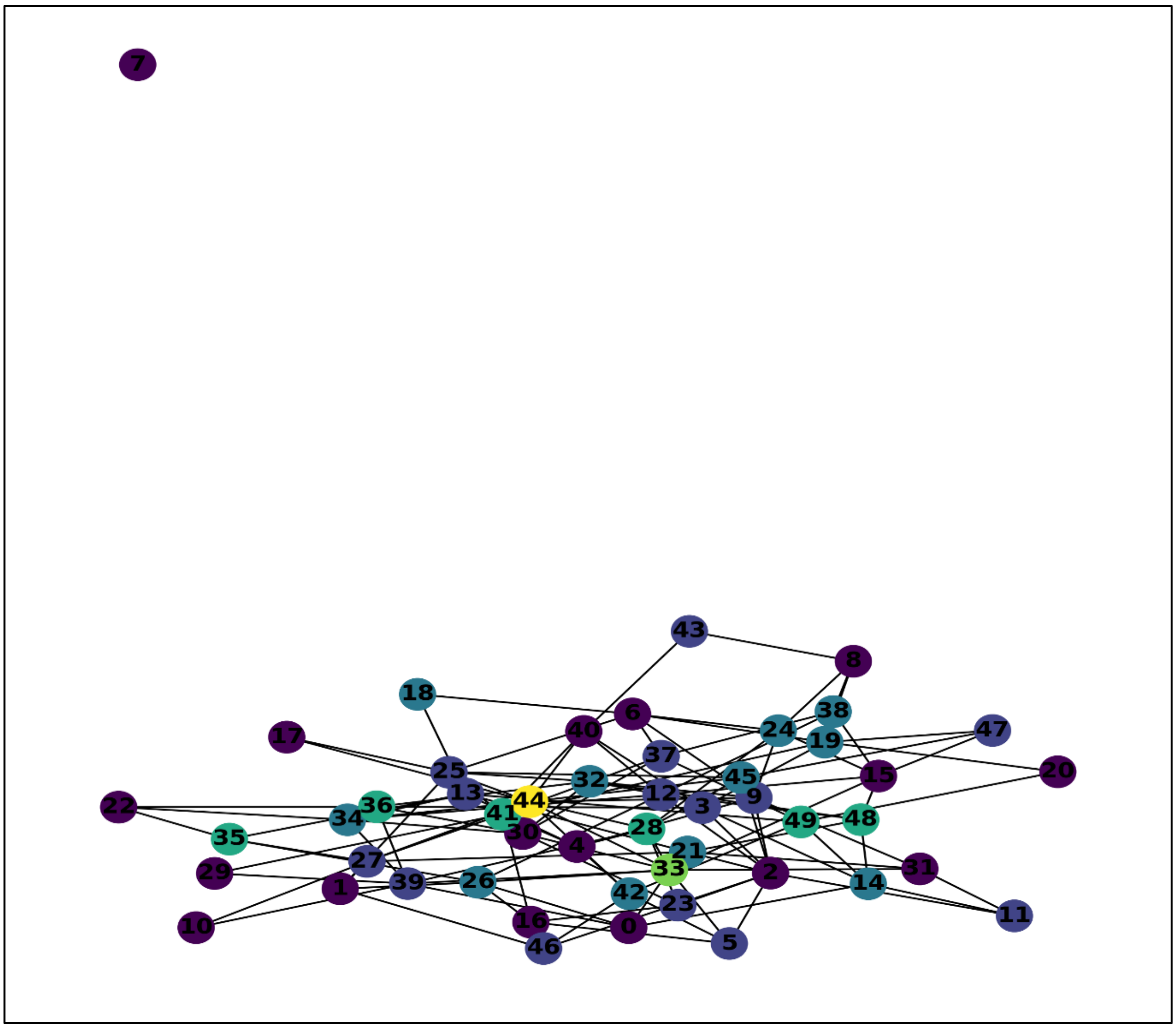


Table 3: Distributed Graph Coloring Results – 50 Nodes

Node	Color	Node	Color	Node	Color	Node	Color
0	0	14	2	28	3	42	2
1	0	15	0	29	0	43	1
2	0	16	0	30	0	44	5
3	1	17	0	31	0	45	2
4	0	18	2	32	2	46	1
5	1	19	2	33	4	47	1
6	0	20	0	34	2	48	3
7	0	21	2	35	3	49	3
8	0	22	0	36	3	Valid	True
9	1	23	1	37	1	k	6
10	0	24	2	38	2	Execution Time (s)	0.005305
11	1	25	1	39	1		
12	1	26	2	40	0		
13	1	27	1	41	3		

4.4 Using Nodes 75

The Bacterial Foraging Optimization (BFO) algorithm was evaluated for its effectiveness in solving the graph coloring problem on a 75-node graph. As illustrated in Figure 4 and detailed in Table 4, the algorithm successfully assigned colors such that no two adjacent nodes shared the same color, verified by the "Valid: True" entry. The algorithm efficiently utilized only 6 distinct colors, showcasing its capability to minimize chromatic numbers. The nodes exhibited a balanced color distribution, with swift execution, completing the task in just 0.006367 seconds. This underscores the algorithm's computational efficiency and accuracy, making it suitable for real-time applications and larger datasets. These results affirm the BFO algorithm's promise in addressing graph coloring challenges, though future research should explore its scalability and robustness in more complex graph structures.

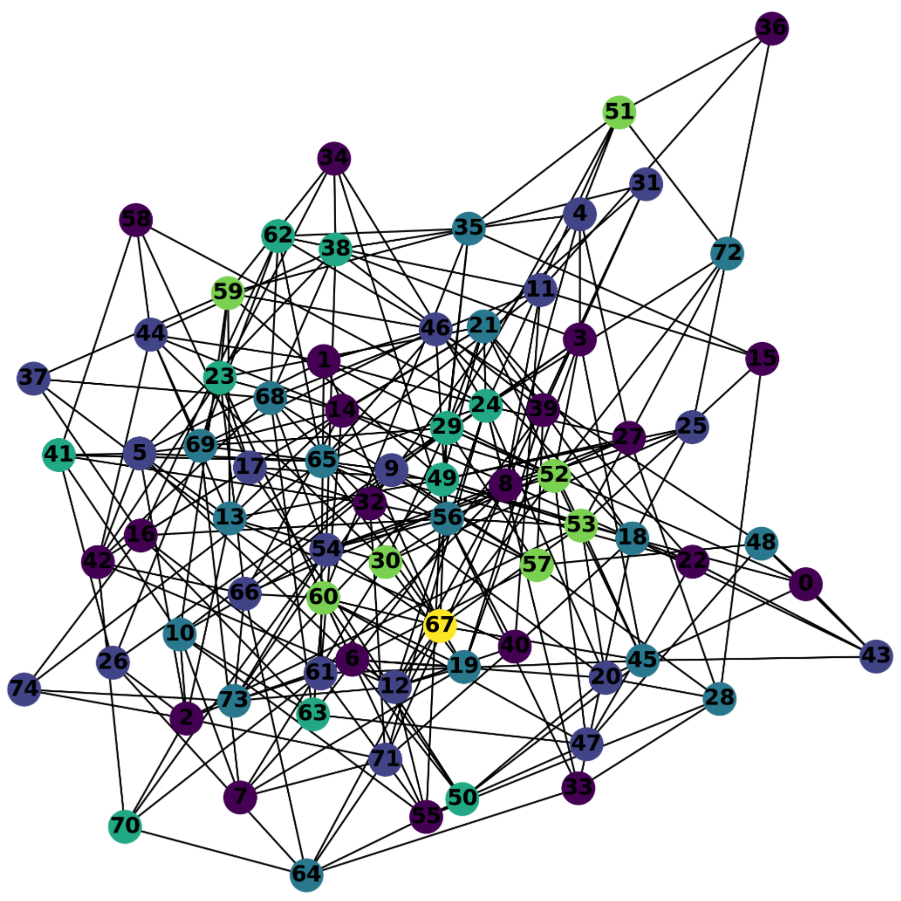


Fig 4: Graph Coloring Results – 75 Nodes

Table 4: Distributed Graph Coloring Results – 75 Nodes

Node	Color	Node	Color	Node	Color	Node	Color	Node	Color
0	0	16	0	32	0	48	2	64	2
1	0	17	1	33	0	49	3	65	2
2	0	18	2	34	0	50	3	66	1
3	0	19	2	35	2	51	4	67	5
4	1	20	1	36	0	52	4	68	2
5	1	21	2	37	1	53	4	69	2
6	0	22	0	38	3	54	1	70	3

7	0	23	3	39	0	55	0	71	1
8	0	24	3	40	0	56	2	72	2
9	1	25	1	41	3	57	4	73	2
10	2	26	1	42	0	58	0	74	1
11	1	27	0	43	1	59	4	Valid	True
12	1	28	2	44	1	60	4	k	6
13	2	29	3	45	2	61	1	Execution Time (s)	0.006367
14	0	30	4	46	1	62	3		
15	0	31	1	47	1	63	3		

4.5 Using Nodes 100

We meticulously evaluated the performance of the Bacterial Foraging Optimization (BFO) algorithm for the graph coloring problem, specifically on a 100-node graph. Our objective was to ensure that each node was assigned a color distinct from its adjacent nodes, thereby adhering to the graph coloring constraints.

The outcomes, depicted in Figure 5 and detailed in Table 5, underscore the efficiency and precision of the BFO algorithm in this context. Table 5 meticulously enumerates each node and its corresponding color, showcasing a strategic distribution that guarantees no two adjacent nodes share the same hue. This valid coloring, corroborated by the "Valid: True" annotation, signifies the algorithm's success in meeting the fundamental criteria of graph coloring.

Remarkably, the BFO algorithm accomplished this feat utilizing merely 7 distinct colors, a testament to its optimization capabilities. This minimal chromatic number reflects the algorithm's adeptness at efficiently partitioning the graph's nodes while avoiding color conflicts. Furthermore, the algorithm executed this task with exceptional speed, completing the entire process in a mere 0.007576 seconds. This rapid execution highlights the BFO algorithm's computational efficiency and its potential for scalability to larger, more complex graphs.

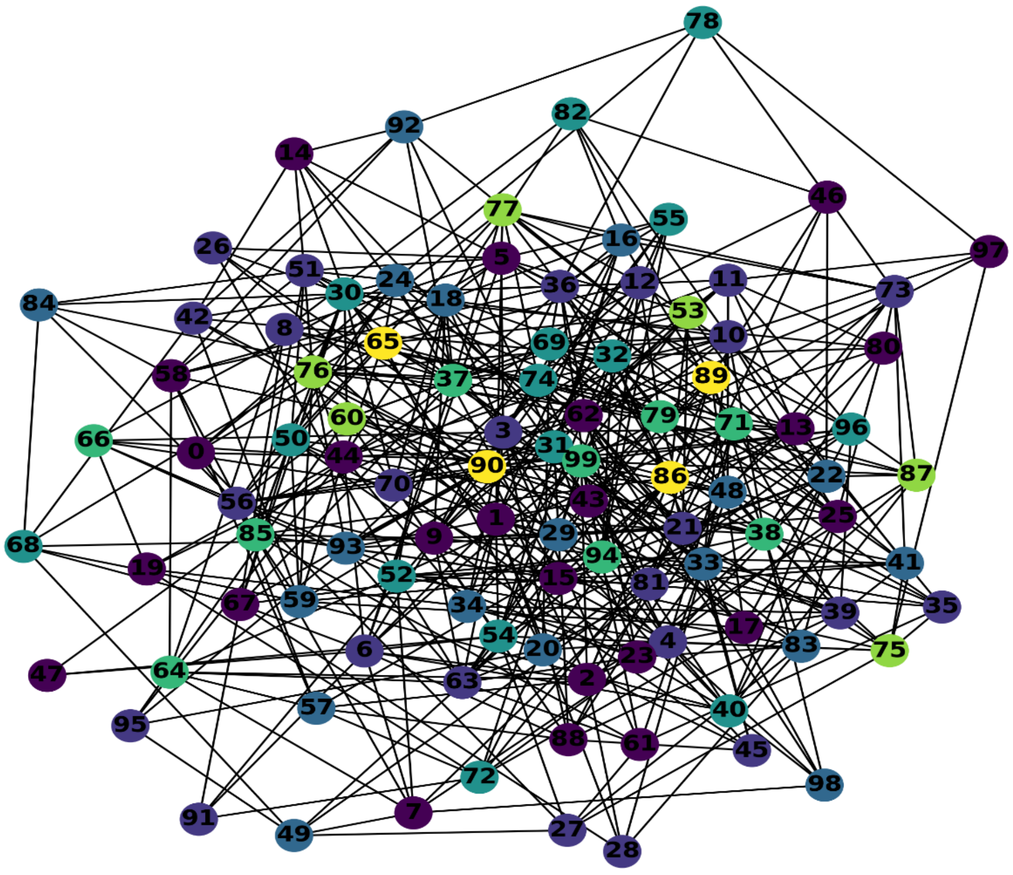


Fig 5: Graph Coloring Results – 100 Nodes

Table 5: Distributed Graph Coloring Results – 100 Nodes

Node	Color	Node	Color	Node	Color	Node	Color	Node	Color
0	0	21	1	42	1	63	1	84	2
1	0	22	2	43	0	64	4	85	4
2	0	23	0	44	0	65	6	86	6
3	1	24	2	45	1	66	4	87	5
4	1	25	0	46	0	67	0	88	0
5	0	26	1	47	0	68	3	89	6
6	1	27	1	48	2	69	3	90	6
7	0	28	1	49	2	70	1	91	1
8	1	29	2	50	3	71	4	92	2
9	0	30	3	51	1	72	3	93	2
10	1	31	3	52	3	73	1	94	4
11	1	32	3	53	5	74	3	95	1
12	1	33	2	54	3	75	5	96	3
13	0	34	2	55	3	76	5	97	0
14	0	35	1	56	1	77	5	98	2
15	0	36	1	57	2	78	3	99	4

16	2	37	4	58	0	79	4	Valid	True
17	0	38	4	59	2	80	0	k	7
18	2	39	1	60	5	81	1	Execution Time (s)	0.007576
19	0	40	3	61	0	82	3		
20	2	41	2	62	0	83	2		

5. Conclusion

Overall, applying the Bacterial Foraging Optimization (BFO) method to graphs with 10 to 100 nodes has yielded useful insights into its efficacy for solving the graph coloring problem. During the studies, the BFO algorithm repeatedly showed its ability to allocate distinct colors to nodes while ensuring that neighboring nodes were not assigned the same hue. The algorithm successfully applied 7 unique colors to the 10-node graph, resulting in a valid coloring. This outstanding work was completed in a mere 0.004464 seconds. This emphasizes its strong computational efficiency and skill in handling smaller graph structures with low computational burden. When the method was expanded to 25 nodes, it continued to perform efficiently, utilizing only 3 unique colors and finishing the task quickly in 0.005134 seconds. The efficiency of the algorithm was demonstrated in the 50-node graph, as it used 6 colors and completed the operation in 0.005305 seconds. Although faced with increased intricacies in the 75-node and 100-node networks, the BFO method remained resilient. The 75-node graph was successfully colored using 6 colors in 0.006367 seconds, indicating steady performance and guaranteeing an even distribution of colors. Similarly, the method employed 7 colors on the 100-node network, achieving completion in 0.007576 seconds, while ensuring that no two adjacent nodes shared the same color. The results highlight the adaptability and effectiveness of the BFO algorithm in handling graph coloring tasks of different graph sizes. The technique demonstrated exceptional performance in smaller to medium sized graphs, executing quickly and using little colors. It also showed scalability and dependability in bigger graphs, but it required slightly more CPU resources. These findings indicate potential avenues for future research to enhance the method, especially in terms of adapting it for larger and more complex graph architectures. To summarize, the Bacterial Foraging Optimization algorithm provides a convincing resolution to the graph coloring problem, combining computing efficiency and accuracy. The fact that it may be successfully applied to graphs of various sizes highlights its potential usefulness in actual applications that require efficient graph analysis. Further study should prioritize improving the scalability and adaptability of the technology, expanding its usefulness in real-world situations that need intricate graph calculations.

6. References

1. Wang, Xiangyu, Xueming Yan, and Yaochu Jin. "A graph neural network with negative message passing and uniformity maximization for graph coloring." *Complex & Intelligent Systems* 10.3 (2024): 4445-4455.
2. Shainky, And Asha Ambhaikar. “Optimizing Graph Colorings: Unleashing the Power of Heuristic Algorithms for Enhanced Problem-Solving.” pp 249–262 In: Tanwar, S., Singh, P.K., Ganzha, M., Epiphaniou, G. (eds) Proceedings of Fifth International Conference on Computing, Communications, and Cyber-Security. IC4S 2023. Lecture Notes in Networks and Systems, vol 991. Springer, Singapore. (2024) https://doi.org/10.1007/978-981-97-2550-2_19
3. Kawakami, Soma, et al. "Ising-machine-based solver for constrained graph coloring problems." *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 107.1 (2024): 38-51.
4. Cummins, Chase, and Richard Veras. "Reinforcement Learning for Graph Coloring: Understanding the Power and Limits of Non-Label Invariant Representations." *arXiv preprint arXiv:2401.12470* (2024).
5. Hu, Yanjuan, et al. "Optimal selection of cloud manufacturing resources based on bacteria foraging optimization." *International Journal of Computer Integrated Manufacturing* 37.1-2 (2024): 165-182.
6. Singh, Law Kumar, et al. "Emperor penguin optimization algorithm-and bacterial foraging optimization algorithm-based novel feature selection approach for glaucoma classification from fundus images." *Soft Computing* 28.3 (2024): 2431-2467.
7. Malhotra, Karan, et al. "A solution to graph coloring problem using genetic algorithm." *EAI Endorsed Transactions on Scalable Information Systems* (2024).

8. Brighen, Assia, et al. "A new distributed graph coloring algorithm for large graphs." *Cluster Computing* 27.1 (2024): 875-891.
9. Kamal, Jihan Sabilla, et al. "The Use of Graph Coloring Theory to Complete Lecture Scheduling at Kuningan University."
10. Barenboim, Leonid, and Michael Elkin. "Distributed graph coloring: Fundamentals and recent developments." (2022).
11. Mishra, Samaresh, and Itishree Swain. "Exploring the Applications of the Bacterial Foraging Optimization Algorithm (BFOA) for the Graph Coloring Problem."
12. Malik, Kiran, Sunita Choudhary, and Jyoti Malik. "Bacterial Foraging Optimization Algorithm (BFOA) for Graph Coloring Problem: A Review." *International Journal of Computer Science & Management Studies* 13.4 (2013).