International Journal of Innovative Research in Computer Science and Technology (IJIRCST)
ISSN: 2347-5552, Volume-12, Issue-4, July 2024
https://doi.org/10.55524/ijircst.2024.12.4.1
Article ID IRP1464, Pages 1-9
www.ijircst.org

A Comparative Study of Cat Swarm Algorithm for Graph Coloring Problem: Convergence Analysis and Performance Evaluation

Ayesha Saeed[1], Ali Husnain[2], Anam Zahoor[3], and Rashad Mehmood Gondal[4]

[1] Research Assistant, Department of Computer Science, University of Chicago, USA
[2,3,4] Research Assistant, Department of Computer Science, University of Lahore, Lahore, Pakistan

Correspondence should be addressed to Ayesha Saeed; ayesha.saeed98@yahoo.com

Received: 16 May 2024 Revised: 31 May 2024 Accepted: 14 June 2024

ABSTRACT- The Graph Coloring Problem (GCP) is a significant optimization challenge widely suitable to solve scheduling problems. Its goal is to specify the minimum colors (k) required to color a graph properly. Due to its NP-completeness, exact algorithms become impractical for graphs exceeding 100 vertices. As a result, approximation algorithms have gained prominence for tackling large-scale instances. In this context, the Cat Swarm algorithm, a novel population-based metaheuristic in the domain of swarm intelligence, has demonstrated promising convergence properties compared to other population-based algorithms. This research focuses on designing and implementing the Cat Swarm algorithm to address the GCP. By conducting a comparative study with established algorithms, our investigation revolves around quantifying the minimum value of k required by the Cat Swarm algorithm for each graph instance. The evaluation metrics include the algorithm's running time in seconds, success rate, and the mean count of iterations or assessments required to reach a goal.

KEYWORDS- Cat Swarm Optimization, Cat Swarm Algorithm, Graph Coloring, Graph Coloring Problem

I. INTRODUCTION

In the past century, the graph coloring problem has gained popularity among researchers and has emerged as a highly explored optimization problem The popularity of GCP naturally arises due to its numerous practical applications, including scheduling, timetabling, and frequent assignment. To add these, other significant uses also include aircraft flight, scheduling, and bioprocesses task allocation. These problems can be modeled into graphs and thus be solved in the same manner as GCP. [1] GCP has been resolved by Optimization algorithms in past decades. These include Genetic Algorithms, Stimulated Annealing and Colony optimization, and artificial bee colony.

Optimization algorithms randomly search for solutions guided by some hints to find the next solution. Due to the randomness of the solution found, optimization algorithms employ fitness functions to help them decide whether a given solution is good and how good the solution is compared to other solutions. The fitness function serves as a guide to the algorithm in finding the optimal solution. [8] With the success of other optimization algorithms like the artificial bee colony algorithm and ant colony optimization in solving GCP, this study finds motivation in exploring the feasibility of another optimization algorithm in solving the same problem and, at the same time determining how fitness function affects the efficiency of an algorithm.

Based on the common behaviors of cats, a relatively new optimization algorithm is proposed. This algorithm is known as cat swam optimization [8][12][20] and claims better performance than another popular optimization algorithm which is the particle swam optimization (PSO), and its improved variant, PSO with weighting factor. Previous research has been done using cat swam optimization in wall-solving Sudoku puzzles and clustering. Now it will be helpful in finding a solution for the graph coloring issue.

II. LITERATURE REVIEWS

In this research article the application of two novel heuristic algorithms derived from previously studied techniques are explored. The first algorithm is a modified version of the Largest Degree Ordering (LDO) algorithm, while the second algorithm is a transformation of the Saturation Degree Ordering (SDO) algorithm. In the suggested modifications, Algorithm 1 introduces a combined approach of the modified Largest Degree Ordering (LDO) algorithm and the Incident Degree Ordering (IDO) technique. This algorithm follows the basic principles of LDO but incorporates IDO when faced with nodes of equal degree, aiding in the selection process. On the other hand, Algorithm 2 combines the Saturated Degree Ordering (SDO) and the Largest Degree Ordering (LDO) methods. It operates similarly to SDO, but when confronted with nodes of the same degree, it employs LDO to make the appropriate choice. Comparative analysis reveals that Algorithm 1, the modification of LDO, outperforms the original LDO algorithm in terms of the total count of colors utilized. Similarly, Algorithm 2, the modification of SDO, surpasses the performance of SDO by employing the minimum colors needed for graph coloring. [1]

This study introduces a pair of heuristic approaches for solving graph coloring problem, which involve modifications or improvements to existing techniques. One algorithm combines the Largest Degree Ordering (LDO) with the Incident Degree Ordering (IDO), while the next combines the Saturated Degree Ordering (SDO) with the

Largest Degree Ordering. A comparative evaluation is conducted to evaluate the effectiveness of these modified techniques and compare them with existing methods. [11]

In this research paper, an investigation was conducted to explore the application of a three-heuristics approach for graph coloring, specifically focusing on the largest set of non-adjacent vertices. The algorithm employed in the study is referred to as the Min_Max algorithm, which involves the sequential selection of a vertex with the lowest degree followed by a vertex with the highest degree. To analyze its effectiveness, the Min_Max approach was applied, tested, and compared against two alternative approaches: the selection of a vertex with lowest degree first (SNMD) and the selection of a vertex with highest degree first (SNXD). The comparison was carried out considering both CPU time and graph size under varying densities. The results obtained from the comparison highlight the superiority of the Min_Max algorithm and SNXD over SNMD in terms of several factors, including the time required to obtain the first maximum independent set, CPU execution duration, and the count of nodes that were successfully colored. These findings demonstrate the reliability and effectivity of the Min_Max approach and SNXD in graph coloring tasks when compared to the SNMD approach. [2]

In this research paper, a machine learning-based approach is proposed for the selection of robotic algorithms aimed at solving the graph coloring problem (GCP). To this end, a comprehensive set of 78 attributes relevant to the GCP is identified, which can be computed efficiently and have a significant impact on solving the problem. The performance of six meta-heuristics is assessed for solving the GCP. The collected data is then utilized to train various classification algorithms, enabling the prediction of the algorithm with the highest expected performance on new instances. To improve the performance of the machine learning algorithms, the influence of parameters is examined, and different techniques for data discretization and feature selection are evaluated. Experimental evaluations are carried out to evaluate the effectiveness of the heuristics. The results demonstrate that a solver incorporating machine learning techniques achieves substantially superior performance corresponded to any individual heuristic approach. [17]

This research paper delves into the exploration of deterministic graph coloring algorithms, specifically focusing on contraction and sequential types. Sequential algorithms can be enhanced through the integration of backtracking, resulting in relatively efficient approaches for determining the coloring count of a graph. Furthermore, incomplete backtracking techniques are employed, leading to the development of novel heuristics for graph coloring. Through extensive experimentation, unexpected results are obtained, shedding light on the performance and effectiveness of the investigated algorithms. These findings contribute to advancing the understanding and improvement of deterministic graph coloring methods, highlighting the potential for innovative approaches and heuristics in solving graph coloring problems. [15]

This research paper introduces a new graph coloring algorithm and compares it to various existing algorithms. The algorithm demonstrates efficient O(n^2) time complexity for most scattered graphs, making it well-suited for large-scale scheduling problems. The Recursive Largest First (RLF) algorithm yields the fastest results in the conducted experiments, exhibiting unique O(n^2) time complexity for low-edge density graphs. Moreover, the RLF procedure provides a standardized method for generating test data, enabling the construction of large graphs with known chromatic numbers using minimal parameters. Overall, the proposed algorithm shows promise for effective graph coloring, particularly in the context of scheduling problems. [16]

This study introduces a new technique for graph coloring problem which is called Welsh Powel technique. In this technique the vertices are sorted on the basis of its valance. Then the vertices are compared largest valance vertex is assign to a specific color. This technique does not always provide optimal solution but is a good solution for graph coloring problem. By using Welsh Powel technique approximately three colors are used which is an optimal answer. [18]

This research paper delves into the classical graph coloring problem, providing an overview of its fundamental components such as chromatic numbers, k coloring factor, and the number of vertices. The study also explores the Greedy algorithm as a means to achieve optimal solutions for graph coloring. Moreover, the paper examines the applications of the graph coloring problem in well-known domains such as the Sudoku game and map coloring. Extensive demonstrations of the graph coloring problem are presented to showcase its utility in solving these applications. Additionally, it is highlighted that the Greedy algorithm has the potential to enhance these applications further. [7]

This research paper introduces Cat Swarm Optimization (CSO), a fresh algorithm derived from the behaviors of cats. CSO consists of pair of sub-models, seeking and tracing, which emulate cat behaviors. Experimental findings using six test functions illustrate that CSO outperforms Particle Swarm Optimization (PSO). Unlike previous findings where PSO with a weighting factor showed faster convergence, CSO exhibits even better performance. This highlights the effectiveness of CSO and its potential as an advanced swarm intelligence technique. [6]

This research paper discusses Data Clustering and introduces a new optimization technique called the blind, naked mole-rats (BNMR) approach for this purpose. Data grouping aims to divide a dataset into classes without prior knowledge of inter-class relations. The famous K-means algorithm is commonly used for clustering. The BNMR algorithm outperforms other methods by achieving higher accuracy in fewer iterations and with reduced computational time and cost. It offers a promising approach for efficient and effective data clustering. [18]

This research paper focuses on two popular optimization algorithms, Ant Colony Optimization and Particle Swarm Optimization (PSO), which simulate the behaviors of specific creatures. In addition, a novel optimization algorithm called Cat Swarm Optimization (CSO) is introduced, leveraging the behaviors of cats as a model. The research elaborate explanation of the CSO algorithm, including its pair of sub-models: the Seeking Mode and the Tracing Mode. Furthermore, a comparative analysis between CSO and PSO-type algorithms is conducted to assess their respective performance. Finally, the paper concludes with a summary of the findings. [20]

This research paper presents an innovative technique called Adaptive Dynamic Cat Swarm Optimization (ADCSO). The approach introduces a dynamic inertia weight and acceleration coefficient to augment the algorithm's abilities in Investigation and utilization. Experimental results on six benchmark problems demonstrate that ADCSO outperforms the original CSO algorithm. It achieves superior solution quality, converges faster, and demonstrates better fitness values. ADCSO proves to be an effective enhancement to CSO for optimization tasks.[12]

The research paper demonstrates the effectiveness of Cat Swarm Optimization (CSO) in solving the Graph Coloring Problem (GCP), successfully solving all instances. CSO can find optimal solutions for specific graph types, with the characteristics of the resolution influenced by the chosen fitness function. The paper also investigates the use of Iterated Local Search (ILS) for GCP, comparing the performance of different algorithms. Ongoing experiments on larger instances from the benchmark suite will be presented at the Symposium, providing additional insights into algorithm effectiveness and scalability. [9]

III. RESEARCG METHOLOGY

The research methodology employed in this study focuses on the Graph Coloring Problem (GCP), which includes allocating colors to the vertices of a graph while adhering to specific conditions [1][2][3]. The primary constraint in GCP is that adjacent vertices should not be allocated the identical color, as this would lead to conflicts. An adequately colored graph is one in which there are no conflicts among adjacent vertices. In addition to achieving proper coloring, GCP aims to reduce the colors used. Therefore, the main purpose of GCP is to find the optimal coloring for a graph while utilizing the fewest colors possible. The chromatic number is the least count of colors required to color a graph properly.

Finding a solution for a Graph Coloring Problem (GCP) could lead to two types of solutions. Firstly, a complete solution refers to any proper graph coloring where adjacent vertices have different colors. Secondly, an optimal solution is obtained when the proper coloring utilizes the minimum count of colors equal to the chromatic number of the graph. [19]

GCP can be solved using many optimization techniques. In optimization, the primary objective is to uncover the most cost-effective or highest-performing alternative that adheres to the imposed constraints. Many algorithms were suggested in the field of optimization. E.g., Ant colony optimization [8][12], Particle swarm optimization [6][8], Particle swarm optimization, and Simulated Annealing. [8] Several of these proposed algorithms utilized swarm intelligence. Several algorithms proposed in this study draw inspiration from swarm intelligence, which focuses on the coordination and self-organization of large groups of individuals, both natural and artificial. Swarm intelligence involves decentralized control and relies on the group's collective behavior, where local interactions among individuals and their environment play a crucial role [6][8][12].

Swarm intelligence exhibits distinct properties that contribute to its unique nature:

- It encompasses a large number of individuals.
- The individuals within the swarm exhibit relative homogeneity, sharing identical characteristics or belonging to a limited number of topologies.
- Interactions among entities are governed by fundamental behavioral principles, primarily relying on nearby data that directly influences the individual or is acquired through environmental factors.
- The collective behavior and overall system dynamics emerge from the interactions among individuals and their environment, showcasing the self-organizing nature of swarm behavior.

We are choosing experimental research methodology. Experimental methodology is the most used methodology for the optimization problem. Experimental evaluation typically encompasses two distinct phases. The initial exploratory phase involves collecting measurements to identify critical aspects and generate relevant questions regarding the system under evaluation. Subsequently, the evaluation phase is dedicated to addressing these identified questions. A meticulously designed experiment begins by outlining the questions the experiment aims to answer.[13]

A. Cat Swarm Optimization for k-GCP

Cat Swarm Optimization (CSO) is derived from the careful observation of cat behaviors and consists of a pair of distinct sub-models: the Tracing Mode and the Seeking Mode. These sub-models are specifically designed to emulate and replicate the behaviors exhibited by cats during different phases of their activities. [6][8][12][20] These behaviors are the resting behavior and the actively moving or chasing behavior. They are named as the Seeking mode and the Tracing mode respectively. Similar to other optimization algorithms, a cat in Cat Swarm Optimization (CSO) is associated with a position and velocity. The position of a cat shows a potential outcome within the optimization problem being addressed. [6][8][12][20]
The two common traits of cats are The resting behavior and The actively moving behavior. The cats wait for a long time and observe the surroundings carefully to move towards the next destination.

- ***Behavior of cats***

In Cat Swarm Optimization (CSO), each cat is defined by several characteristics. These include its position, which shows a promising outcome within the optimization problem. Additionally, each cat maintains information about its velocity, the best position discovered so far, and a flag indicating whether it is currently in tracing or seeking mode. These attributes collectively contribute to the overall behavior and performance of the CSO algorithm.
Velocity: Present the velocity that applies to the cat position.
Position: Present as a current position of the cat.
Flag: Use to determine if the cat is in search mode or tracking mode.
The two sub modes of the cat swarm optimization algorithm are built based on these two prominent behaviors of cats which are seeking mode for resting and tracing mode for moving.

- ***Seeking Mode***

During rest periods, a cat remains alert and observant, carefully scanning its environment in search of the optimal following location to move from its current position. This attentive behavior allows the cat to assess its surroundings and make informed decisions about subsequent movements.

Seeking mode has four parameters.

i. Seeking Memory Pool (SMP)

- **ii.** Seeking Range Of Dimension (SRD)
- **iii.** Count of Dimension to Change (CDC)
- **iv.** Self Position Consideration (SPC)

i) Seeking Memory Pool (SMP)

In Cat Swarm Optimization (CSO), the Seeking Memory Pool (SMP) has a vital part in evaluating the seeking behavior of each cat. The SMP refers to the amount of the memory and stores copies of potential solution points. Every cat utilizes the information in the SMP to evaluate and select the best spot to transfer to. The selection process follows predefined rules and considers several observations from the memory pool, enabling the cat to make informed decisions during optimization.

ii) Seeking Range Of Dimension (SRD)

The Seeking Range Declaration (SRD) parameter determines the mutation rate for specified dimensions within Cat Swarm Optimization. In the seeking mode, when a dimension is chosen for mutation, the discrepancy between the new and old values is constrained within a predefined range specified by SRD. This range restriction ensures the updated values remain within acceptable bounds during the solution update process. SRD plays a important part in maintaining the feasibility and effectiveness of the optimization approach.

iii) Count of dimension to change (CDC)

CDC determines the count of varying dimensions in Cat Swarm Optimization (CSO). It controls diversity and exploration during optimization, impacting search behavior and convergence. It also plays important role in updating the solution.

iv) Self-Position Consideration (SPC)

CPC is a Boolean variable used in Cat Swarm Optimization (CSO) to determine whether the present posture of a cat will be considered eligible for movement. The value of CPC can be either true or false and does not impact the Seeking Memory Pool (SMP). CPC allows cats to decide whether to move or remain stationary. If a cat identifies its current position as the optimal solution, it will choose not to transfer and remain in its current spot.
The operation of seeking mode can be presented in below 5 steps:

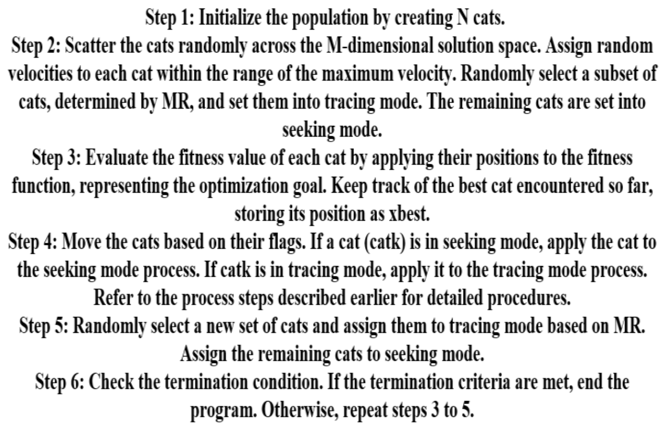
- ***Tracing Mode***

This is the mode where the cat has to attack or move. It moves according to its velocity. The position on which the cat moves to any position from its current position is according to its velocity. The new solution is determined by updating the current solution using the value of velocity. This process is known as process of exploration in which a wide search is used throughout the solution space. It makes the algorithm robust and does not get trap in local optimum.

- ***General Cat Swarm Optimization(CSO)***

In the previous subsection, we introduced two sub-models within the Cat Swarm Optimization (CSO) algorithm: the seeking and tracing modes. To incorporate both methods into the algorithm, we introduce a mixture ratio (MR) that determines the balance between the seeking and tracing modes.
By closely observing the movements of cats, we have noted that they dedicated a significant portion of their awake time resting. During this resting period, they move cautiously and slowly, sometimes even staying in their original position. To incorporate this behavior into CSO, we utilize the seeking mode. The behavior of cats chasing the targets is captured by the tracing mode. Evidently, the MR value should be small to ensure that maximum time is spent in the seeking mode, closely resembling real-world behavior.
The CSO process can be outlined in six sequential steps as follows:

The General CSO Algorithm


Step 1: Initialize the population by creating N cats.
Step 2: Scatter the cats randomly across the M-dimensional solution space. Assign random velocities to each cat within the range of the maximum velocity. Randomly select a subset of cats, determined by MR, and set them into tracing mode. The remaining cats are set into seeking mode.
Step 3: Evaluate the fitness value of each cat by applying their positions to the fitness function, representing the optimization goal. Keep track of the best cat encountered so far, storing its position as xbest.
Step 4: Move the cats based on their flags. If a cat (catk) is in seeking mode, apply the cat to the seeking mode process. If catk is in tracing mode, apply it to the tracing mode process. Refer to the process steps described earlier for detailed procedures.
Step 5: Randomly select a new set of cats and assign them to tracing mode based on MR. Assign the remaining cats to seeking mode.
Step 6: Check the termination condition. If the termination criteria are met, end the program. Otherwise, repeat steps 3 to 5.

B. Chromatics: Solving GCP using CSO

In solving GCP, we can consider any number of colors to use. However, it is a more suitable option to narrow down the colors to consider so that the solution space will also be narrowed down. But on the other hand, if the choice of color is narrowed down too much it could happen that the count of colors being considered is not sufficient to color the graph properly thus it would be impossible to solve GCP.
In order to have a reasonable Maximum limit for the count of colors to consider that ensures a complete solution for a graph; Brook's theorem is utilized in this research. The theorem guarantees that the highest count of colors necessary to color a certain graph is equal to the graph's Brook's value or the maximum-vertex degree of the graph +1.
In CSO the solution of problem is represented by the cat position, thus when solving GCP, the position of the cats represents the problem solution thus when solving GCP, the position of the cats would correspond to the different colorings of the graph. In simple terms what happens in the CSO process is that the cats move from place to place until it finds the best position.
Since GCP has two types of solutions, the approach used is to find a complete solution quickly and gradually optimize the complete solution by reducing the count of colors used until the optimal approach is found. However, to escape being entangled in the local optima, regressions would be allowed at some point through exploration of other solutions. [8]

i) The Chromatics Seeking Mode Process

In this approach, seeking mode is designed to search a complete solution and further optimize it to search the optimal solution. In order to search a complete solution quickly; seeking mode uses a color selection which is biased towards lessening the count of conflicts in the graph.
To lessen the count of conflicts, the selection of colors will be biased towards colors which are not yet assigned to any vertex. Since the color selected is not yet used then this ensures that using it would introduce no conflicts. However,

in the case that there is no more unused color available, the selection would just be done randomly.
When a complete solution is already found, seeking mode would now optimize the solution by lessening the number of colors used. This is done by choosing a vertex from a graph and just copying the color assigned to it. This means that colors which are more commonly used in the graph has a higher probability of being selected and eventually they will replace the less commonly used colors thus reducing the number of colors used. [8]
To better help in finding a complete solution quickly and also to maintain the completeness of a solution while reducing the number of colors a color change restriction is imposed. In color change restriction, the color update is restricted in such a way that only changes which would result in better colorings or lesser conflicts are allowed. This is done by counting the number of conflicts a certain vertex has before and after the color change. If the change resulted to more conflicts then it will be reverted otherwise it will be retained. [8]

ii) The Chromatics Tracing Mode Process

The process in seeking mode ensures finding an equal or better solution from the current solution because of the color change restriction it imposes however this also presents a problem of being trapped in local optima solutions. To avoid such situations tracing mode is implemented without any restriction thus it is free to explore other solutions ever if the solutions found introduce the number of colors used by using a color selection similar to seeking mode when a complete solution is already found. [8]

iii) Fitness Function

Many fitness functions have been discussed in research. We have chosen and implemented the following one.

$$\text{fit }(g)=\frac{1}{1+\Sigma\ (|\Sigma ki = 1Vi|)}$$

Where E(Vi) is he set of conflicts in the ith color class. A color class is simply a group of vertices assigned with a specific color. [8]

iv) Output

The output of the program is the best coloring found by the algorithm [8]. See the below figure 1, figure 2 and figure 3.

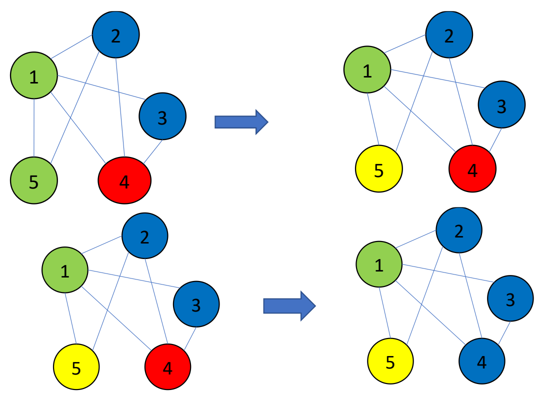
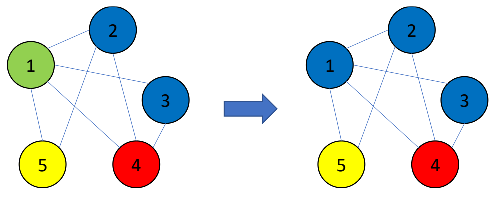
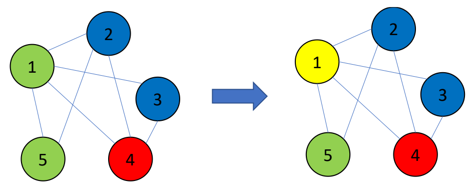


Figure 1: Optimization Step: Selecting a new color for v4



Conflicts of v1 = 0 Conflicts of v1 = 2

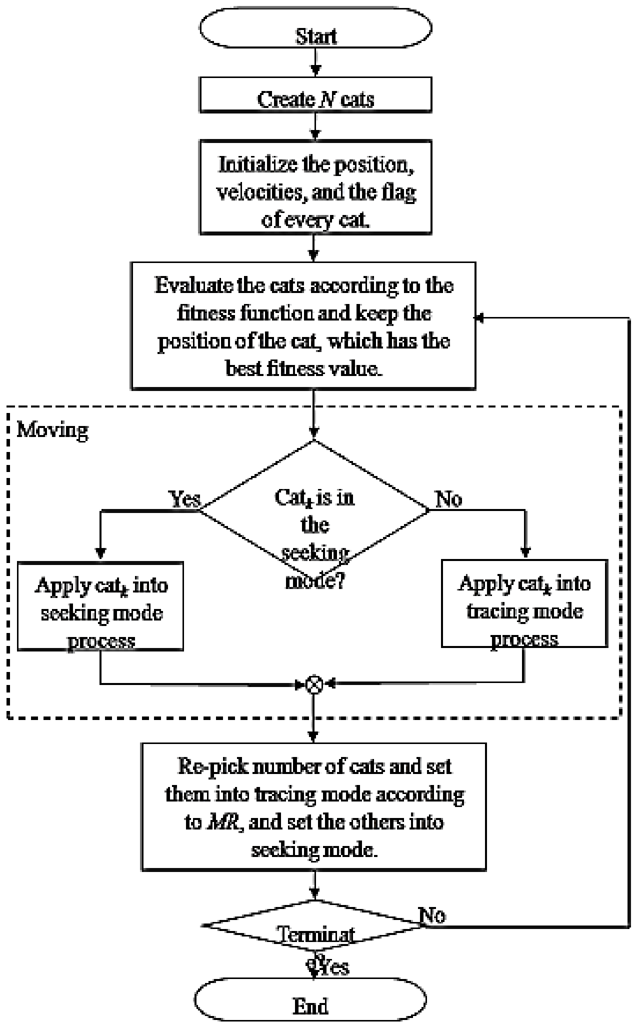
Figure 2: Reverting Step: Color Change in v1



Conflicts of v1=0 Conflicts of v1 = 0

Figure 3: Retaining Step: Color change in v1

Figure 4 and figure 5 is showing the flow chart of CSO algorithm.



Start
Create N cats
Initialize the position, velocities, and the flag of every cat.
Evaluate the cats according to the fitness function and keep the position of the cat, which has the best fitness value.
Moving
Yes
Cat$_k$ is in the seeking mode?
No
Apply cat$_k$ into seeking mode process
Apply cat$_k$ into tracing mode process
Re-pick number of cats and set them into tracing mode according to MR, and set the others into seeking mode.
Terminate
No
Yes
End

Figure 4: General CSO Algorithm Flow Chart

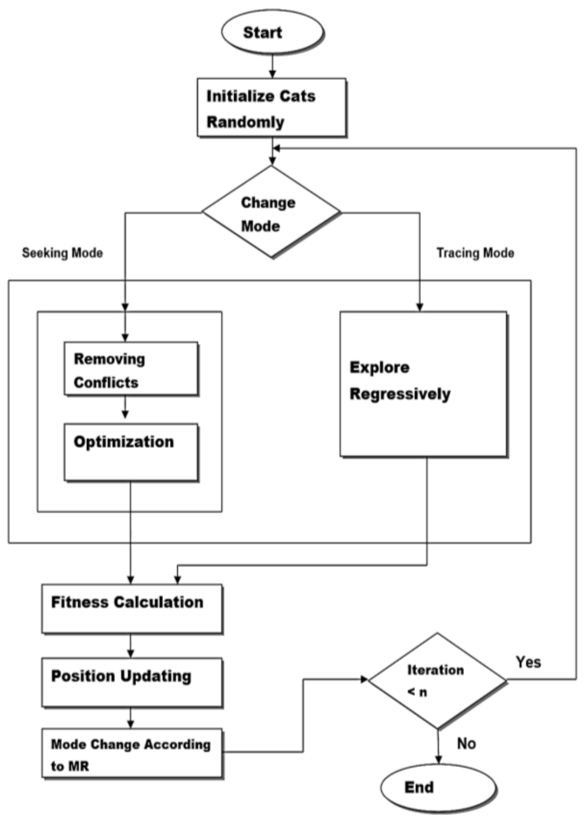


Figure 5: General CSO Algorithm Flow Chart

IV. EXPERIMENTS AND RESULTS

This portion concludes the results projected from our comparative study. It contains the graphs and benchmarks that will show the results gained from the research.

A. *Heuristic graph coloring algorithms*

Our study will primarily revolve around several heuristic graph coloring algorithms, namely FF, FDO, SDO, and LDO. The field of graph coloring encompasses numerous sequential techniques that aim to achieve optimal color assignments. Among these techniques is the Greedy Graph Coloring approach, which places a strong emphasis on the meticulous selection of the next vertex to be colored. In the following subsections, we will delve into the details of the first fit and degree-based ordering approaches, providing a comprehensive explanation of their principles and methodologies.

i) First Fit

In the realm of greedy coloring heuristics, the First Fit algorithm stands out as both the simplest and fastest technique. This algorithm follows a sequential approach, assigning each vertex the lowest available color that satisfies the constraint of not conflicting with neighboring vertices. Notably, the First Fit algorithm boasts the advantages of simplicity and efficiency, with a time complexity of O(n) for its implementation [1][2]

ii) Degree Based Ordering

The proposed strategy offers an improved approach for graph coloring by employing a specific choice standard when determining the next vertex to be colored [1][2] This approach outperforms the First Fit algorithm, which selects vertices in an arbitrary order. In this study, various approaches to choose the subsequent node to be colored have been proposed and explored, aiming to enhance the efficiency and effectiveness of the graph coloring process.

iii) Largest Degree Ordering (LDO)

The Largest Degree Ordering (LDO) algorithm selects a node with the maximum count of neighbors as the next one to be colored. Intuitively, this approach leads to improved graph coloring outcomes compared to the First Fit algorithm. LDO offers the advantage of a more effective color assignment strategy. Implementing the LDO heuristic typically requires a time complexity of O(n^2) [1][2].

iv) Saturation Degree Ordering (SDO)

The Saturation Degree Ordering (SDO) algorithm assigns a saturation degree to each node, which represents the count of its subsequent vertices with different colors. This heuristic offers improved graph coloring outcomes compared to the Largest Degree Ordering (LDO) algorithm. Notably, the SDO algorithm can be implemented with a time complexity of O(n^3) [1][2].

v) Incidence Degree Ordering (IDO)

An alternative version of the Saturation Degree Ordering (SDO) heuristic is the Incidence Degree Ordering (IDO) heuristic. The incidence degree of a vertex is determined by counting the count of its subsequent vertices that have already been colored. This modification offers a different approach to graph coloring and can be implemented with a time complexity of O(n^2) [1][2].

vi) Color Walk

It is a new technique introduced, which involves assigning a random color to a vertex and coloring the graph while adhering to the conditions of the Graph Coloring Problem (GCP). The technique follows a process where adjacent nodes are left uncolored while ensuring that the entire graph is properly colored. Subsequently, another color is randomly selected and the process is repeated. This approach aims to gain graph coloring with the minimum colors used to satisfy the GCP conditions.
The collected data is organized in the form of charts, representing different types of graph instances. Each chart presents a specific set of graph instances, allowing for a comprehensive analysis.

B. *Full Insertion Graphs*

The approach involves a generalization of the Mycielski transformation, where additional nodes are inserted to increase the size of the graph without affecting its density. Following table 1 is the data set of the Full Insertions Graph Instances showing the various techniques implemented and compared:

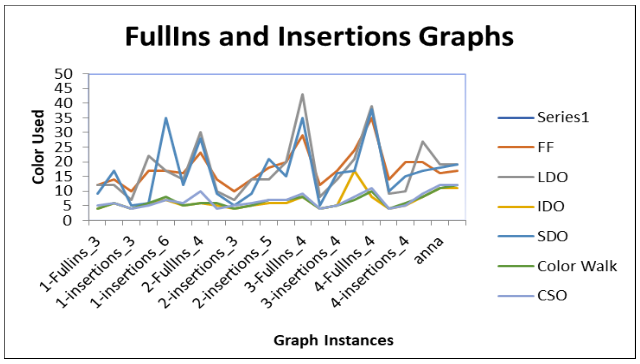


Figure 6: Full Insertion Graphs

Table 1: Full Insertions Graphs

Graph Instances	FF	LDO	IDO	SDO	Col Walk	CSO
	colr used	colr used	colr used	colr used	colr used	colr used
1-Fullins_3	12	12	4	9	4	5
1-insersions_5	14	12	6	17	6	6
1-insertions_3	10	7	4	5	4	4
1-insertions_4	17	22	6	6	6	5
1-insertions_6	17	17	7	35	8	7
2-FullIns_3	16	14	5	12	5	6
2-FullIns_4	23	30	6	28	6	10
2-FullIns_5	14	10	5	9	6	4
2-insertions_3	10	7	4	5	4	5
2-insertions_4	14	14	5	9	5	6
2-insertions_5	18	14	6	21	7	7
3-FullIns_3	20	20	6	15	7	7
3-FullIns_4	29	43	8	35	8	9
3-insertions_3	12	8	4	5	4	4

Above figure 6 is showing the comparison chart of Full Insertion Graphs:

C. Random Graphs

The instances used in this study are derived from Johnson's approach, where a graph is constructed by considering each potential edge [u,v] with a possibility p for every pair of vertices in a given set. The chromatic number of these instances remains unknown. Following table 2 is the data set of the Random Graph Instances showing the various techniques implemented and compared:

Table 2: Random Graphs

Graph Instances	FF	LDO	IDO	SDO	Col Walk	CSO
	colr used	Colr used	colr used	colr used	Colr used	colr used
DSJC125.1	21	20	7	25	7	8
DSJC125.5	69	73	25	74	25	26
DSJC125.9	111	109	54	122	58	56
DSJC250.1	41	41	12	44	13	13
DSJC250.5	144	144	40	160	44	43
DSJC250.9	233	230	95	223	97	100
DSJC500.1	80	83	18	75	19	19
DSJC500.5	284	293	71	258	71	74
fpsol2. i.1	76	84	65	50	65	65
fpsol2.i.2	51	53	30	50	30	32
fpsol2. i.3	49	53	30	48	31	32
games120	19	18	9	15	9	9
homer	27	31	13	25	14	16
huck	17	13	11	15	11	11
inithx.i.1	66	68	54	89	54	54
inithx.i.2	85	49	31	52	31	31
inithx.i.3	39	49	31	51	81	31
jean	14	20	10	16	10	10

Figure 7 is showing the comparison chart of random graphs.

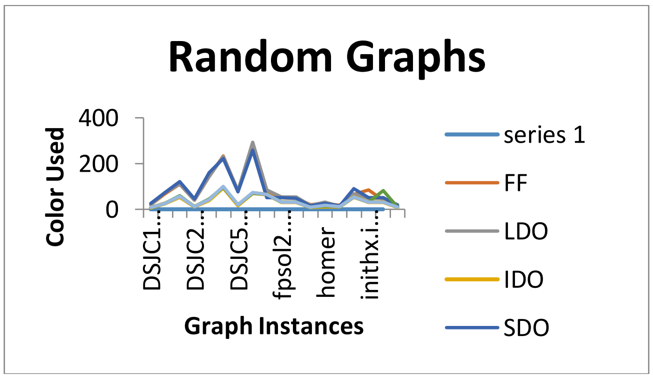


Figure 7: Random Graphs

Results

The analysis of the graph instances employed in the Random Graphs experiment reveals a discernible pattern. Particularly, when dealing with large graphs, there is a tendency to utilize the maximum number of colors.

D. Leighton Graphs

The graphs utilized in this research are specifically generated using a procedure considering the count of vertices, the preferred chromatic number, the average vertex degree, and a random vector of integers. This procedure creates a specified number of cliques within the graph structure. Notably, the known chromatic number of these instances can be readily ascertained. Following table 3 is the data set of the Leighton Graph Instances showing the various techniques implemented and compared:

Table 3: Leighton Graphs

Graph Instances	FF	LDO	IDO	SDO	ColWalk	CSO
	Col use	Col use	Col use	Col use	Col use	Col use
le450_15a	68	66	19	66	22	22
le450_15b	63	76	18	67	42	23
le450_15c	113	114	27	123	31	32
le450_15d	116	26	32	105	32	30
le450_25a	76	77	25	76	29	28
le450_25b	74	74	30	71	28	28
le450_25c	131	128	31	121	36	37
le450_25d	120	131	29	122	38	35
le450_5a	54	48	12	42	13	14
le450_5b	48	47	11	48	12	13
le450_5c	81	78	12	81	16	17
le450_5d	80	77	13	72	17	16

Figure 8 is showing the comparison chart of Leighton graphs.

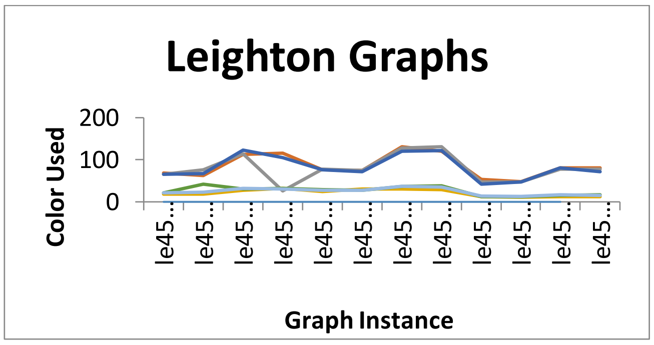


Figure 8: Leighton Graphs

Results

The findings from the analysis of Leighton Graphs exhibit distinct characteristics. Specifically, the IDO, CSO, and Color Walk algorithms demonstrate promising results by utilizing fewer colors. In contrast, the FF, LDO, and SDO algorithms yield more unpredictable outcomes.

E. Queen Graphs

The queen graph, derived from Donald Knuth's Stanford Graph Base, is constructed based on an n x n chessboard. In this graph, each square of the board corresponds to a node, resulting in a total of n^2 nodes. Although the exact chromatic number of the queen graph remains unknown, it is worth noting that the maximum clique in the graph is at most n. Therefore, the chromatic number is not less than n. Following is the data set of the Queen Graph Instances showing the various techniques implemented and compared:

Table 4: Queen Graphs

Graph Instances	FF	LDO	IDO	SDO	ColWalk	CSO
	Colused	Colused	Colused	Colused	Colused	Colused
queen5_5	13	16	7	17	7	7
queen6_6	16	23	10	21	9	10
queen7_7	19	26	12	30	11	11
queen8_8	22	39	15	32	12	13
queen8_12	26	49	15	45	15	16
queen9_9	25	48	15	35	14	13
queen10_10	28	67	17	41	16	17
queen11_11	31	72	18	46	16	17
queen12_12	34	99	20	46	18	18
queen13_13	37	93	22	62	20	19
queen14_14	40	120	24	61	21	21
queen15_15	43	127	26	84	22	22
queen16_16	46	163	26	80	24	25

Figure 9 is showing the comparison chart of Queen graphs.

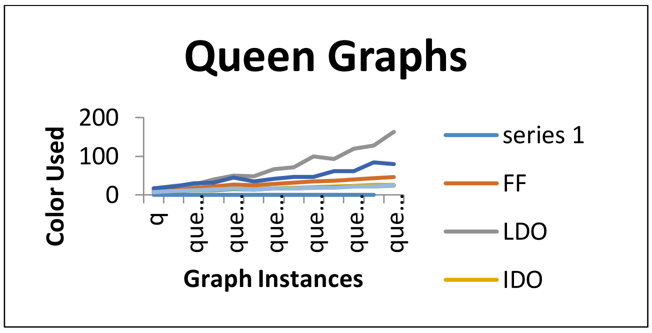


Figure 9: Queen Graphs

Results

The analysis of Queen Graphs reveals a fascinating pattern wherein the count of vertices and the count of colors used exhibit a consistent increase. As the size of the graph increases, there is a corresponding rise in the count of colors required for proper coloring, demonstrating a clear and consistent trend.

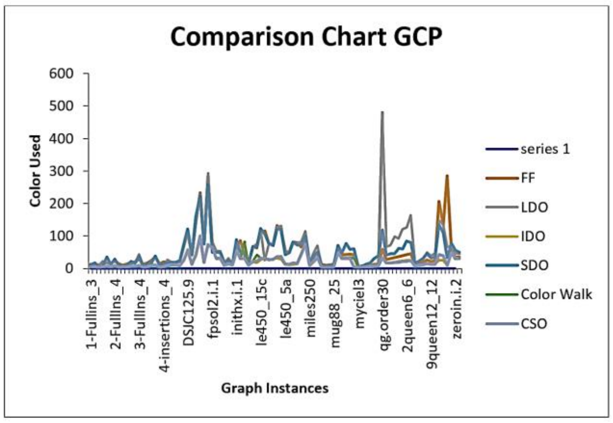
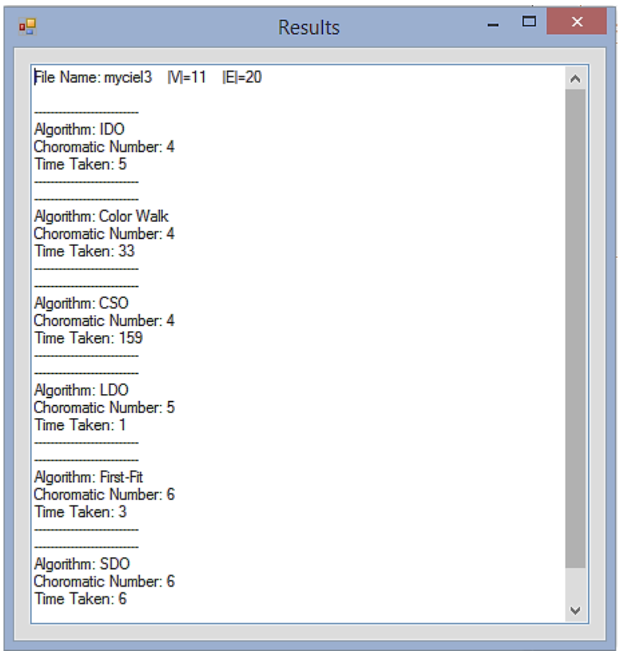


Figure 10: Overall Comparison



```
File Name: myciel3   |V|=11   |E|=20

--------------------------
Algorithm: IDO
Choromatic Number: 4
Time Taken: 5
--------------------------

--------------------------
Algorithm: Color Walk
Choromatic Number: 4
Time Taken: 33
--------------------------

--------------------------
Algorithm: CSO
Choromatic Number: 4
Time Taken: 159
--------------------------

--------------------------
Algorithm: LDO
Choromatic Number: 5
Time Taken: 1
--------------------------

--------------------------
Algorithm: First-Fit
Choromatic Number: 6
Time Taken: 3
--------------------------

--------------------------
Algorithm: SDO
Choromatic Number: 6
Time Taken: 6
```

Figure 11: Comparison between All Techniques

F. The Overall Comparison

The results of the study are presented in a comprehensive comparison chart (figure 10 and figure 11), showcasing the performance of various heuristic graph coloring algorithms across all graph instances.

V. CONCLUSION

After performing a comprehensive analysis of the entire dataset of graph instances, it is evident that the meta-heuristics employed in our research have yielded significant results. Specifically, the Cat Swarm Optimization (CSO) Algorithm has consistently outperformed other algorithms, providing the most optimal solutions. However, it is important to note that the drawback of CSO lies in its time-consuming nature, as it requires a substantial amount of time to complete its cycle.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

1. H. Al-Omari and K. E. Sabri, "New graph coloring algorithms," American Journal of Mathematics and Statistics, vol. 2, no. 4, pp. 739-741, 2006. Available from: https://thescipub.com/pdf/jmssp.2006.439.441.pdf
2. H. Almara'beh and A. Suleiman, "Heuristic algorithm for graph coloring based on maximum independent set," Journal of Applied Computer Science & Mathematics, vol. 13, no. 6, pp. 19-23, 2012. Available from: https://www.jacsm.ro/view/?pid=13_3
3. R. Dorne and J. K. Hao, "A new genetic local search algorithm for graph coloring," in Parallel Problem Solving from Nature—PPSN V: 5th International Conference Amsterdam, The Netherlands September 27–30, 1998 Proceedings 5, vol. 5, Springer Berlin Heidelberg, 1998, pp. 745-754.D. Available from: https://doi.org/10.1007/BFb0056916
4. Bre1az, "New Methods to Color the Vertices of a Graph," Management Science/Operations Research, Communications

of the ACM, vol. 22, no. 4, pp. page numbers, 1979. Available from: https://doi.org/10.1145/359094.359101

5. J. A. Torkestani, "A New Vertex Coloring Algorithm Based On Variable Action-Set Learning Automata," Computing and Informatics, vol. 29, pp. 447–466, Arak, Iran, 2010. Available from: https://www.researchgate.net/publication/220106403_A_New_Vertex_Coloring_Algorithm_Based_on_Variable_Aaction-Set_Learning_Automata
6. S. C. Chu, P. W. Tsai, and J. S. Pan, "Cat swarm optimization," in PRICAI 2006: Trends in Artificial Intelligence: 9th Pacific Rim International Conference on Artificial Intelligence Guilin, China, August 7-11, 2006 Proceedings 9, Springer Berlin Heidelberg, 2006, pp. 854-858. Available from: https://www.geeksforgeeks.org/cat-swarm-optimization/
7. L. Rouse and M. Albrandt, "A Touch of Color: Graph Coloring," Mini-Excursion 2. Available from: https://www.claytonschools.net/cms/lib/MO01000419/Centricity/Domain/241/ChromaticChapter.pdf
8. N. D. Bacarisas and J. P. T. Yusiong, "The Effect of Varying the Fitness Function on the Efficiency of the Cat Swarm Optimization algorithm in Solving the Graph Colouring Problem," Annals. Computer Science Series, vol. 9, 2011. Available from: https://www.sciencedirect.com/topics/computer-science/swarm-optimization-algorithm
9. M. Chiarandini and T. Stützle, "An application of iterated local search to graph coloring problem," in Proc. Computational Symposium on Graph Coloring and its Generalizations, New York, USA, Ithaca, Sep. 2002, pp. 112-125. Available from: https://imada.sdu.dk/u/marco/Publications/Files/gcp-ils-Ithaca.pdf
10. A Kosowski and K. Manuszewski, "Classical coloring of graphs," Contemporary Mathematics, vol. 352, pp. 1-20, 2004. Available from: https://umv.science.upjs.sk/madaras/ATG/coloring%20algorithms.pdf
11. Mansuri, V. Gupta, and R. S. Chandel, "Coloring Programs in Graph Theory," Int. Journal of Math. Analysis, vol. 4, no. 50, pp. 2473-2479, Bhopal, (M.P.) India, 2010. Available from: https://www.geeksforgeeks.org/graph-coloring-applications/
12. M. Orouskhani, Y. Orouskhani, M. Mansouri, and M. Teshnehlab, "A Novel Cat Swarm Optimization Algorithm for Unconstrained Optimization Problems," I.J. Information Technology and Computer Science, vol. 11, pp. 32-41, 2013. Available from: https://www.mecs-press.org/ijitcs/ijitcs-v5-n11/IJITCS-V5-N11-4.pdf
13. R. Elio, J. Hoover, I. Nikolaidis, M. Salavatipour, L. Stewart, and K. Wong, "About computing science research methodology," in Google Scholar Google Scholar Reference, 2011. Available from: https://www.semanticscholar.org/paper/About-Computing-Science-Research-Methodology-Elio-Hoover/55a528f0f10b4ff4d4ec40d8fc30460ecf51f697
14. J. R. Allwright, R. Bordawekar, P. D. Coddington, K. Dincer, and C. L. Martin, "A comparison of parallel graph coloring algorithms," SCCS-666, pp. 1-19, 1995. Available from: https://www.researchgate.net/publication/2296563_A_Comparison_of_Parallel_Graph_Coloring_Algorithms
15. W. Matula, G. Marble, and J. D. Isaacson, "Graph coloring algorithms," in Graph Theory and Computing, New York, NY, USA: Academic Press, 1972, pp. 109-122. Available from: https://www.geeksforgeeks.org/graph-coloring-applications/
16. T. Leighton, "A graph coloring algorithm for large scheduling problems," Journal of Research of the National Bureau of Standards, vol. 84, no. 6, p. 489, 1979. Available from: https://nvlpubs.nist.gov/nistpubs/jres/84/jresv84n6p489_a1b.pdf
17. N. Musliu and M. Schwengerer, "Algorithm selection for the graph coloring problem," in International Conference on Learning and Intelligent Optimization, Berlin, Germany, Jan. 2013, pp. 389-403. Available from: https://link.springer.com/chapter/10.1007/978-3-642-44973-4_42
18. "Vertex Coloring: Welsh Powell Algorithm," Module 2: Planning and Scheduling, pp. 53-55. Available from: http://mrsleblancsmath.pbworks.com/w/file/fetch/46119304/vertex%20coloring%20algorithm.pdf
19. M. Taherdangkoo, M. H. Shirzadi, M. Yazdi, and M. HadiBagheri, "A Robust Clustering Method Based on Blind, Naked Mole-Rats (BNMR) Algorithm," Swarm and Evolutionary Computation, vol. 10, pp. 1–11, 2013. Available from: https://www.sciencedirect.com/science/article/abs/pii/S2210650213000035
20. S. C. Chu and P. W. Tsai, "Computational intelligence based on the behavior of cats," International Journal of Innovative Computing, Information and Control, vol. 3, no. 1, pp. 163-173, 2007. Available from: https://www.researchgate.net/publication/228721750_Computational_intelligence_based_on_the_behavior_of_cats