Hybridization of Genetic Algorithm with Bitstream Neurons for Graph Coloring

Timir Maitra[1], Anindya J. Pal[1], Tai-hoon Kim[2,*], Debnath Bhattacharyya[2]

[1]*Heritage Institute of Technology,*
Kolkata-700107, India.
timirmaitra@gmail.com, anindyajp@yahoo.com

[2]*Department of Multimedia,*
Hannam University,
Daejeon, Korea.
taihoonn@empal.com,debnathb@gmail.com

Abstract:

The population based approach of GA allows large jumps in the search space. However GAs has not proved successful for graph coloring because of the large degree of symmetry of the solution space. In fact, because of this symmetry mismatch it is very unlikely to produce a fit offspring when combining two good solutions. Thus GAs are often considered on inappropriate approach for problems such as graph coloring with a highly degenerate objective function. In order to compensate for this degeneracy we have applied Bitstream neuron (Boltzmann Machine) to the solution obtained from GA. Unlike traditional approaches of GA and ANN the proposed hybrid algorithm is guaranteed to have 100% convergence rate to valid solution with no parameter tuning. Experiments of such a hybrid algorithm are carried out on large DIMACS Challenge benchmark graphs. Results prove very competitive. Analysis of the behavior of the algorithm sheds light on ways to further improvement.

1. Introduction

Proper coloring of a simple, symmetric and connected graph is a classical problem. Here we are dealing with m–colorability optimization problem. The problem asks for the smallest integer m for which the graph G = (V, E) with vertex set V and edge set E, can be colored properly. This minimum integer is called the chromatic number $\chi(G)$. A k-vertex coloring of any graph G is an assignment of k–colors 1, 2, 3,……k to the vertices of G.

The reasons why the graph coloring problem is important, are twofold. First, there are several areas of practical interest, in which the ability to color an undirected graph with a small number of colors as possible, has direct influence on how efficiently a certain target problem can be solved. Such areas include, timetable scheduling [1], examination scheduling [2], register allocation [3], printed circuit testing [4], electronic bandwidth allocation [5], microcode optimization [6], channel routing [7],

the design and operation of flexible manufacturing systems [8], computation of sparse Jacobian elements by finite differencing in mathematical programming [9], etc.
The other reason is that, the graph coloring problem has been shown to be computationally hard at a variety of levels: not only its decision problem variant is NP-complete [10], but also its approximate version is NP-hard [11].
These two reasons are important enough to justify the quest for heuristics to solve graph coloring problem, and to pace it together with the favorite problems on which new metaheuristics are tested.
Heuristics in optimization is any method that finds an 'acceptable' feasible solution in reasonable time. Local search procedure based heuristics most of the time terminates with local optima. Randomization and restarting approaches is in need to overcome the local optima problem. Among the existing heuristic approaches for the graph coloring we propose a hybrid algorithm of Boltzmann Machine of Artificial Neural Network with Genetic Algorithm (GA) to implement. Both of this method is very suitable to implement randomization and restarting approaches. Genetic algorithm is an evolutionary approach which does not have the above said problem but it cannot guaranty the optimal solution. Hopfield neural network the most common and generally used artificial neural network itself has local optimum problem that is the reason to select Bitstream neuron model of ANN.
The purpose of the selection of GA is the power of GA to give different partial or probable solutions. GA itself or with some modification can solve GCP. GA were introduced not to solve a particular problem but to investigate the effects of natural adaptation in stochastic search algorithms. It consists of a population of possible problem solutions that get refined over time through selection, crossover and mutation operator. Fitness function has a crucial role in GA that evaluates the quality of the solution calculating fitness of a solution. GA evaluates a population of N solutions; it also evaluates a much larger number of partial solutions. It does this job without spending more than N fitness function evaluations. The general scheme of a GA is discussed in different papers.
In Shawe-Taylor and Zerovnik[12] introduced the Generalized Boltzmann Machine (GBM) as an extension to the Boltzmann machine that enables us to map constraint problems, requiring more than two states, onto a recurrent neural network. In this paper, we extend the use of bitstreams from the bi-polar, stochastically connected Boltzmann machine to a recurrent network of stochastically connected multi-state bit stream neuronsWe initiate Bitstream with the outcome of GA. The job of the ANN is to refine the solution to the desired level. The ANN is characterized by its pattern of connections between the neurons (called its architecture), its method of determining the weights on the connections (called its training or learning algorithm), and its activation function. There are several areas where ANN is already in use. Few to mention are Function approximation including fitness approximation, classification

including pattern and sequence recognition and sequential decision making, data processing including filtering clustering, robotics etc. There are several models that include ANN, and we select Boltzmann Machine (BM). The Boltzmann distribution has some beautiful mathematical properties and it is intimately related to information theory. In particular, the difference in the log probabilities of two global states is just their energy difference. The equilibrium distribution is independent of the path followed in reaching equilibrium are what make Boltzmann machines interesting.

2. Previous work

If we look into previous work on heuristic, we find metaheuristic approach for search where particle swarm algorithm used. The particle swarm algorithm adjusts the trajectories of a population of particles through a problem space on the basis of information about each particle's previous best performance and the best previous performance of its neighbors. The proposed paper is a reworking of the algorithm to operate on discrete binary variables. In the binary version trajectories are changes in the probability that coordinate will take on zero or one value. Trajectories are defined as changes in position on some number of dimensions [13]. The way ant colonies function suggests an approach to stochastic combinatorial optimization. The characteristics of the proposed approach are positive feedback accounts for rapid discovery of good solutions; distributed computation avoids premature convergence; and the greedy heuristic helps find acceptable solutions in the early stages of search process. Traveling salesman problem solved by this approach also compared with Tabu search and Simulated annealing [14]. The work presents a heuristic technique that combines Learning Automata (LA) with Random Walk (RW) for solving the decision variant of GCP. The approach is Traditional random walk with LA based learning capability encoding the GCP as a Boolean Satisfiability Problem. The experiments demonstrate that the Learning Automata significantly improve the performance of RW [15]. Q'tron Neural Network for GCP is one of the approaches that built as a known-energy system. It can make Local-minima-free and perform the goal-directed search. Implementation is on DIMACS through continuous refinement of the goal [16]. The work adopted multilevel cooperative search heuristic to solve the graph coloring problem. Multilevel cooperative search is a meta-heuristic based on information shared among concurrently executing search algorithms and different representation levels of the solution space. Information sharing and the multilevel representation combined with frequent re-starts of the local search algorithms can be used to compensate for the lack of gradient in cost functions. Re-starts are based of graph coloring solutions projected from a high level representation of the solution space to a lower level. This search technique is a parallel algorithm, one algorithm per

level of representation [17]. Multiple-restart quasi-Hopfield network is also been proposed for this problem where only problem-specific knowledge is embedded in the energy function that the algorithm tries to minimize. It is an up-gradation of their previous work on this field. They used k-state neurons instead of binary neurons, that reduces the number of connections in the network [18]. A model called the Integer Merge Model has been proposed which aims to reduce the time complexity of graph coloring algorithms. They also claim that the model provide information that will help to create heuristics that works faster. They used it in the complete algorithm DSATUR, and in two version of an incomplete approximation algorithm; an evolutionary algorithm and the same evolutionary algorithm extended with guiding heuristics. Integer Merge Operation attempts to minimize the number of color if two nodes are not connected by a normal edge or a hyper edge [19]. The proposed DNA algorithm based on surfaces for the graph coloring problem is presented which is easy to implement and error-resistant as they claim. According to their approach first whole combinatorial color assignments to the vertices of a graph are synthesized and immobilized on a surface; then a vertex is legally colored while those adjacent to it with illegal colors are deleted; and the cycle is repeated until finally the correct color assignments to the graph are reached [20].

Genetic algorithm proved to be a good approach where large pools of trivial solutions are created satisfying the constraints of the problem. We found different implementation of GA on graph coloring during past long years. A proposed work shows a stochastic search by a genetic algorithm is employed to find a near optimal solution for the node partitioning problem that includes graph coloring problem. The solution representation by elegant data structure, together with genetic operations and selection policies employed in the GA procedure are presented. This work does not require the number of disjunct node sets to be given a priori. The real life problem VLSI design is solved through this above mentioned algorithm [21]. Genetic local search algorithm for graph coloring problem is been proposed in another work. The algorithm combines and original crossover based on the notion of union of independent sets and a powerful local search operator (tabu search). This hybrid algorithm allows improving results of some large instances of Dimacs benchmarks [22]. The proposed symmetry breaking approach presents a genetic algorithm that breaks the symmetry of the graph coloring problem by fixing the colors of the nodes in large clique of the graph. They breaks the symmetry by applying GAs in cyclic permutations evolve populations where almost all individuals have the color of pivot node fixed. They claim the feature reduces the search space by one dimension [23]. The proposed hybrid genetic algorithm combines the global search power of GA with local search power of local optimization algorithm. Here each GCP solutions called phenotype is presented isomorphic genotypes conceptually. A metric function

between two phenotypes is defined by the least hamming distance between the corresponding sets of isomorphic genotypes. A new crossover technique named Harmonic Crossover is proposed in this work. The phenotypic distance between two parents is also used to predict promising regions in the problem space. The local optimization algorithm is applied only in the most promising regions restrictedly and intensively [24]. A new evolutionary formulation of the graph coloring problem was introduced based on the interplay between orderings and colorings of vertices. The formulation breaks symmetry in the solution space and provides opportunities for combining evolutionary and other search techniques. The formulation used the relationship between a graph's chromatic number and its acyclic orientations. The algorithm experimented on DIMACS computational challenge graphs [25]. The application of parallel genetic algorithms for solving graph coloring problem with message passing paradigm is being used as an implementation tool in the proposed work. In message passing processes exchange data with one another by sending messages. This method of implementing parallel is very flexible, universal, portable to various architectures and can be highly efficient as claimed. This approach involves execution of an evolutionary algorithm on numerous processing units that improve GA's performance. The parallelization is being achieved by creating number of separate populations which exchange genetic information during migration process [26].

Artificial neural network also used in different way to solve graph coloring problem; we are going to mention few of them. A parallel algorithm based on the McCulloch-Pitts binary neuron model and the Hopfield neural network is one of the proposed work that solves four coloring map problems and k-colorability problems. The work shows the computational energy is always guaranteed to monotonically decrease with the Newton equation. They tested their algorithms through large number of simulation runs [27]. The proposed work presents a new algorithm using a maximum neural network model to k-color vertices of simple undirected graph. They claim the algorithm guaranteed to converge to valid solutions with no parameter tuning needed [28]. A proposed work that efficiently approximate MAX-CLIQUE in a special case of the Hopfield network whose stable states are maximal cliques. The work includes several energy-descent optimizing dynamics; both discrete and continuous. There is comparison with Mean-Field Annealing – an efficient approximation to Simulated Annealing – and a stochastic dynamics. They show all dynamics approximate much better than one which emulates a 'naive' greedy heuristic [29]. In a graph coloring solution using Hopfield Neural Network, they implemented this algorithm using FPGAs, which is defined by VHDL. Their algorithm supports parallel execution. They used Hill Climbing method to avoid Local minima problem that is obvious in Hopfield Neural Network. Their work is good for small graph. Time complexity is not

dependent on the size of graph in most of the cases. But this implementation has a limitation of working with large graph. To work on huge graph they need to implement huge number of neurons that need huge number of gates. They took some randomly generated graph of small size, neither well known standard graph nor big graph that has big number of vertex and edges [30]. A hardware based approach which can overcome the limitation of number of neurons and interconnection required for Hopfield net based approaches. This network uses only N neurons and N^2 weights for N nodes. They used amplifier and comparator in proper way with the help of resistance and capacitance. They applied Kirchhoff's current law to few nodes. The monotonic function used to manipulate operation, different term used to perform different job, one to minimize color values, another to define different coloring of adjacent nodes. Time function is actually Lyapunov function for the network that causes dynamics of the network to converge local minimum from the arbitrary initial state [31].

Previous work on Boltzmann Machine are like the Mean field approach to learning in BM, which minimizes computational complexity of exact algorithm. Change in complexity is from exponential to cubic in the number of neurons. In this approach gradient descent procedure is not being used for learning process instead weights are computed directly from the fixed point equation. It is shown that the result is close to optimal solutions and give significant improvement over the naive mean field approach [32]. Multi State Bitstream Neuron introduced by replacing stochastic activation function with stochastic weights. MSBSN is shown to approximate a Generalised Boltzmann Machine and it is compact and easy for fast hardware implementation. It is a recurrent network of stochastically connected multi-state bit stream neurons [33]. The proposed parallel algorithm for the boltzmann machine is new synchronously parallel algorithms for the BM. This is independent of the connection pattern and the equilibrium distributions are the Gibbs distributions. Solution of the optimization problems is guaranteed [34]. A natural method is to use stochastic weights rather than stochastic activation functions; where the proposed work introduce a new model in which each neuron has very simple functionality but all the weights are stochastic. It is shown that the stationary distribution of the network uniquely exists and it is approximately a Boltzmann- Gibbs distribution when the size of the network is not too small. A new technique to implement simulated annealing is also proposed and the power of network is comparable with that of a Boltzmann Machine [35]. An unsupervised learning algorithm for a stochastic recurrent neural network based on BM architecture is proposed. The maximization of the Mutual Information between the stochastic output neurons and the clamped inputs is used as an unsupervised criterion for training the network. The resulting learning rule contains two terms corresponding to Hebbian and anti-Hebbian learning. Using

that optimal non-linear and recurrent implementation of data compression of boolean patterns are obtained [36]. The proposed work defines a new neuron structure to be implemented on Generalised Boltzmann Machine (GBM). In this neuron stochastic weights replaced the stochastic activation function, they termed this neuron as Multi State Bitstream Neuron (MSBSN). According to their algorithm the Graph Coloring problem maps directly to GBM. They used negative identity matrix converted weight matrix associated with each graph edge. Stopping criteria is energy function and Boltzmann temperature. Their experiments includes generated graph with vertex range 50 to 400, no benchmark graph is used to test the algorithm. They didn't provide any table of result of the experiments. They demand the algorithm is capable of solving different optimization problem and easy to hardware implementation as well [38].

3. Our work

As mentioned earlier our work proposed a way to color a graph satisfying the coloring constraints. To discuss the basic definitions and concepts of Graph, let G=(V,E) be an simple undirected graph where V= {v1,v2,…} is the set of elements called vertices in G, and E={e1,e2,…} is the set of elements called edges in G, each element of the set E can be identified by any unordered pair of vertices. The vertex set of a graph G is referred to as V(G) and its edge set as E(G). Obviously, $V \cap E = \Phi$. An edge {x,y} is usually written as xy. A vertex v is incident with an edge e if $v \in e$. The two vertices incident with an edge are its endvertices. Two vertices connected by an edge are called adjacent vertices and neighbours. In a graph, for an edge if the end vertices are same, then it's called self loop; if more than one edge have same end vertices, then they are called parallel edges. The set of all neighbors of a vertex v in G is the neighborhood of G and is denoted NG(v), or simply N(v). The degree dG(v) = d(v) of a vertex is the number of edges to which it is incident. If G is a graph, then this is equal to the number of neighbors of v. The maximum degree of G is $\Delta(G) = \max \{dG(v) \mid v \in V(G)\}$. The minimum degree of G is $\delta(G) = \min \{dG(v) \mid v \in V(G)\}$. Conventionally, we write δ and Δ instead of $\delta(G)$ and $\Delta(G)$. We are going to consider vertex coloring problem of the graph. Our algorithm starts with input of a graph from text file. Job starts with creation of adjacency matrix from input.

Our algorithm starts with Genetic Algorithm (GA). GA were introduced by Holland in 1970s [20] not to solve a particular problem but to investigate the effects of natural adaptation in stochastic search algorithms. Gas consists of a population of possible problem solutions that get refined over time through selection recombination and mutation. The general scheme of a GA discussed in different papers [15,16]. The general scheme of GA can be given as follows**:**

begin
INITIALIZE population with random candidate solutions;
EVALUATE each candidate;
repeat
SELECT parents;
RECOMBINE pairs of parents;
MUTATE the resulting children;
EVALUATE children;
SELECT individuals for the next generation
until TERMINATION-CONDITION is satisfied
end

The scheme of GA falls in the category of *generate-and-test* algorithms. The evaluation function represents a heuristic estimation of solution quality and the variation and the selection operator drive the search process. The most important components in a GA consist of:

+ representation (definition of individuals)
+ evaluation function (or fitness function)
+ population
+ parent selection mechanism
+ variation operators (crossover and mutation)
+ survivor selection mechanism (replacement)

Encoding/Representation

The application of a genetic algorithm to a problem starts with the encoding. The encoding specifies a mapping that transforms a possible solution to the problem into a structure containing a collection of decision variables that are relevant to the problem at hand. A particular solution to the problem can then be represented by a specific assignment of values to the decision variables. The set of all possible solutions is called the *search space*, and a particular solution represents a point in that search space.

Fitness function

Coming up with an encoding is the first thing that a genetic algorithm user has to do. The next step is to specify a function that can assign a score to any possible solution or structure. The score is a numerical value that indicates how well the particular solution solves the problem 1. Using a biological metaphor, the score is the *fitness* of the individual solution. It represents how well the individual adapts to the environment. In this case, the environment is the search space. The task of the GA is

to discover solutions that have high fitness values among the set of all possible solutions.

Population

The role of the population is to hold possible solutions. A *population* is a multiset of genotypes means a group of chromosomes/individuals. A population thus forms a set of solutions for the problem. Each individual x_i corresponds to a fitness value $f(x_i)$. For example, in maximization problem a large value of fitness allows the better chance of selection for reproduction of the individual and hence is better for surviving in the environment.

Parent Selection Mechanism

The role of *parent selection* (*mating selection*) is to distinguish among individuals based on their quality to allow the better individuals to become parents of the next generation.

Parent selection is *probabilistic*. Thus, high quality individuals get a higher chance to become parents than those with low quality. Nevertheless, low quality individuals are often given a small, but positive chance; otherwise the whole search could become too greedy and get stuck in a local optimum. The effect of selection is to return a probabilistically selected parent. Although this selection procedure is stochastic, it does not imply GA employ a directionless search. The chance of each parent being selected is in some way related to its fitness such as fitness based, rank based, or tournament based selection.

Variation Operators

Once the encoding and the fitness function are specified, the user has to choose selection and genetic operators to evolve new solutions to the problem being solved. The selection operator simulates the "survival-of-the-fittest". There are various mechanisms to implement this operator. In order to explore new solutions, the GA relies mainly on two variation operators: crossover and mutation.

Crossover Operator

A binary variation operator is called *recombination* or *crossover*. This operator merges information from two parent genotypes into one or two offspring genotypes. Crossover works by pairing members of the population and mixing pieces of one solution with pieces of another solution. The original pair of individuals is called the parents, and the resulting pair of individuals is called the children. This operator is typically applied with a high probability and is responsible for most of the search performed by the GA. However, that there are many types of crossover operators. The

key idea is to exchange pieces from one solution with pieces of another solution. There are different types of crossover, and they are one-point, two-point and uniform corssover.

Mutation Operator

A unary variation operator is called *mutation*. It is applied to one genotype and delivers a modified *mutant*, the *child* or *offspring* of it. In general, mutation is supposed to cause a random unbiased change. Mutation has a theoretical role: it can guarantee that the space is connected.

Survivor Selection Mechanism

The role of survivor selection is to distinguish among individuals based on their quality. In GA, the population size is (almost always) constant, thus a choice has to be made on which individuals will be allowed in the next generation. This decision is based on their fitness values, favoring those with higher quality.

The Genetic Algorithm (GAGCP) creates initial population of chromosomes by coloring vertices randomly. Algorithm contains a fitness function that checks fitness of the chromosome that counts number of color used to color the chromosome. Depending on the crossover probability we will implement crossover operator on two random pair of chromosomes to create new offspring. Depending on the mutation probability we will implement mutation operator on randomly selected chromosome. After some iteration of this process we will select some best chromosome based on there fitness values to create a good population to work further. Algorithm is given below:

Algorithm GAGCP

INPUT: Total no. of vertices, total no. of edges and vertices that create edges from file and create adjacency matrix of the graph with number of vertices

Step 1) Create random initial pool of chromosomes (population), using n no. of colors

Step 2) Calculate the fitness of the pool

Step 3) Sort the chromosomes of the pool in ascending order of their fitness value

Step 4) P_{best} = best individual

Step 5) For generation = 1 to max_iteration

Step 5.1) If the crossover probability $p_c <= 0.4$

Step 5.1.1) Perform crossover between any two random pair of chromosomes

Step 5.2) If the mutation probability p_m < = 0.6

Step 5.2.1) Randomly choose one chromosome from the pool

Step 5.2.2) Apply n_mutation to the chromosome

Step 5.3) Calculate the fitness of the pool new offspring generated by crossover and mutation

Step 5.4) Take improved chromosomes for next generation

Step 6) Output the best coloration

In our next phase of algorithm we implement artificial neural network, based on Boltzmann Machine model which we will call ANNGCP. Based on the fitness value we will select chromosomes one by one from the population. A Boltzmann machine, like a Hopfield network, is a network of units with an "energy" defined for the network. It also has binary units, but unlike Hopfield nets, Boltzmann machine units are stochastic. The global energy, E, in a Boltzmann machine is identical in form to that of a Hopfield network:

$$E = -\sum_{i<j} w_{ij}\, s_i\, s_j + \sum_i \theta_i\, s_i$$

Where:

- w_{ij} is the connection strength between unit j and unit i.
- s_i is the state, $s_i \in \{0, 1\}$, of unit i.
- θ_i is the threshold of unit i.

The connections in a Boltzmann machine have two restrictions:

- No unit has a connection with itself.
- $w_{ij} = w_{ji} \qquad \forall i, j$. (All connections are symmetric.)

Thus, the difference in the global energy that results from a single unit i being 0 versus 1, written ΔE_i, is given by:

$$\Delta E_i = \sum_j w_{ij}\, s_j - \theta_i$$

A Boltzmann machine is made up of stochastic units. The probability, p_i of the i-th unit being on is given by:

$$p_i = \frac{1}{1 + \exp(-\frac{\Delta E_i}{T})}$$

where the scalar T is referred to as the temperature of the system. The functional unit of neural network; where all signals processed by bit stream neurons are real values represented by 'stochastic bits streams'. A 'stochastic bit stream' is simply a random binary sequence which has value 1 or 0. It can be used to represent an analog quantity in various ways. A common method is the linear mapping from the quantity to the probability of the bit in stream being 1. We call such a bit stream a Bernoulli sequence. The generation of digital bit streams modulated to convey arbitrary probability values.

A stochastic bit stream neuron is processing unit which carries out very simple operations over its input bit streams. All input bi streams are combined with their corresponding weight bit streams and then the weighted bits are summed up. The final total is compared to a threshold value. If the sum is not less than the threshold the neuron gives an output 1, otherwise 0.

The activation function of a Bit Stream Neuron can be computed under the assumption that the input and weight bits are all independently generated. For every weighted input line i we represent its net contribution by a random variable X_i, which has the following probability density function.

$$f_i(x) = \begin{cases} (|w_i v_i| + w_i v_i)/2 & x = 1 \\ (|w_i v_i| - w_i v_i)/2 & x = -1 \\ 1 - |w_i v_i| & x = 0 \\ 0 & \text{otherwise} \end{cases}$$

Where $w_i \in [-1,1]$ is the corresponding weight value and $v_i \in [0,1]$ the input value.

We need to create two matrixes for Boltzmann Machine is state matrix and another matrix for color to vertex mapping. We will calculate energy of network based on the state matrix that represents states of the vertices at a particular time. This will be considered as the parameter of terminating condition of the iteration. In different phases of the algorithm we introduced stochastic behavior like vertex selection for checking improper coloring, selection of the procedure to make proper coloring, selection of color. We introduced another procedure to minimize the color that is color compaction where minimum color value will be replaced with maximum color value and vice-versa for few repetitions.
We executed GA on this to have a set of possible solution, and then tried to get best coloration from each chromosome of the pool of chromosomes through ANN. Our algorithm is given below:

Algorithm ANNGCP

Step 1) Select a possible solution from the set of coloration created by GAGCP

Step 2) Create a matrix for color to vertex mapping

Step 3) Create state matrix and energy calculation function

Step 4) Repeat until energy is minimized

Step 4.1) Repeat until valid coloration is achieved

Step 4.1.1) Select a vertex randomly

Step 4.1.2) Check adjacent vertex for same color

Step 4.1.2.1) Increment adjacent vertex color depending on probability, (if color value reach the maximum value, initialize with minimum value) or assign a random color to that vertex.

Step 4.1.2.2) Update weight vector, state matrix and matrix for vector color mapping

Step 4.1.2.2) Collect the values that we are replacing

Step 4.2) Count color and energy of new coloration

Step 4.3) Find highest replaced color and replace with selected vertex depending on the probability or replace with new randomly selected color

Step 4.4) Short weight vector

Step 4.5) Repeat color compaction to minimize no. of color

Step 4.5.1) Replace minimum color value of the vertices with highest color value and maximum color value with minimum color value alternatively.

4. Experimental Result

Experimental results we got after executing the program that implements the proposed algorithm on different benchmark graphs. We considered different type of graphs of different complexity level and size. Every graph has some specification and reason to be special, few to mention like Game Graphs which is a graph representing the games played in a college football season can be represented by a graph where the nodes represent each college team. Two teams are connected by an edge if they played each other during the season. Knuth gives the graph for the 1990 college football season. Miles Graphs are those graphs that are similar to geometric graphs in that nodes are placed in space with two nodes connected if they are close enough. These graphs, however, are not random. The nodes represent a set of United States cities and the distance between them is given by road mileage from 1947. These graphs are also due to Knuth. Queen Graphs is described as an n x n chessboard, a queen graph is a graph on n^2 nodes, each corresponding to a square of the board. Two nodes are connected by an edge if the corresponding squares are in the same row, column, or diagonal. Unlike some of the other graphs, the coloring problem on this graph has a natural interpretation: Given such a chessboard, is it possible to place n sets of n queens on the board so that no two queens of the same set are in the same row, column, or diagonal. The answer is yes if and only if the graph has coloring number n. Book Graphs, given in a work of literature, a graph is created where each node represents a character. Two nodes are connected by an edge if the corresponding characters encounter each other in the book. Knuth creates the graphs for five classic works: Tolstoy's Anna Karenina (anna), Dicken's David Copperfield (david), Homer's Iliad (homer), Twain's Huckleberry Finn (huck), and Hugo's Les Mis\'erables (jean).

All these algorithms have been implemented in ANSI C and run on a Xeon 2.4GHz machine with 1GB of RAM running the Linux operating system. We have considered the population size as 10. The no_of_iteration for both the two mutations was set to 150 after trial and error. The question mark (?) given in different places of the Table 1 is to define unavailability of the value. Table 1 contains set of sample result to show the performance of the proposed algorithm.

Table 1. Performance Analysis.

SL. No.	Instances	V	E	X(G)	Best Known	ANNGCP
1	1-FullIns 5.col.b, CAR	282	3247	?	6	6

2	2-FullIns 5.col.b, CAR	852	12201	?	7	11
3	3-FullIns 4.col.b, CAR	405	3524	?	7	7
4	5-FullIns 4.col.b, CAR	1085	11395	?	9	11
5	2-Insertions 3.col.b, CAR	37	72	4	4	4
6	2-Insertions 4.col.b, CAR	149	541	4	5	5
7	3-Insertions 5.col.b, CAR	1406	9695	?	6	6
8	anna.col.b, SGB	138	493	11	11	11
9	David.col.b, SGB	87	406	11	11	11
10	Games120.col.b, SGB	120	638	9	9	9
11	Homer.col.b,SGB	561	1629	13	13	13
12	Huck.col.b, SGB	74	301	11	11	11
13	Jean.col.b,SGB	80	254	10	10	10
14	Miles1000.col.b, SGB	128	3216	42	42	42
15	Queen8_8.col.b, SGB	64	1456	9	9	11
16	Queen7_7.col.b, SGB	49	952	7	7	7

5. Conclusion

Our algorithm gives the competitive result for different type of benchmark graphs. To strengthen conclusions made about the power of the algorithm, it is worth to test it on some other classes of large graphs. The main purpose of this paper is to study hybridization of GA and BM on graph coloring problem. In the future, we intend to refine ANNGCP and apply it to find minimal color of large and small but complex graphs in reasonable time.

References

[1] D.C. Wood, “A technique for coloring a graph applicable to large scale time-tabling problems,” Computer Journal, vol. 12, pp. 317-319, 1969.

[2] F. T. Leighton, “ A graph coloring algorithm for large scheduling problems,” Journal of Research of the National Bureau of Standards, vol. 84, no. 6, pp. 489-505, 1979.

[3] F.C. Chow and J.L. Hennessy, “Register allocation by priority based coloring,” Proceedings of the ACM SIGPLAN 84 symposium on compiler construction Newyork, pp. 222-232, 1984.

[4] M.R. Garey, D.S. Johnson and H.C. So., “An application of graph coloring to printed circuit testing,” IEEE Transactions on circuits and systems, vol. 23, pp. 591-599, 1976.

[5] A. Gamst, “Some lower bounds for class of frequency assignment problems,” IEEE Transactions on Vehicular Technology, vol. 35, no. 1, pp. 8-14, 1986.

[6] Micheli G.D., SYNTHESIS AND OPTIMIZATION OF DIGITAL CIRCUITS. McGraw-Hill, 1994.

[7] S.S. Sarma, R. Mondal and A. Seth, "Some sequential graph coloring algorithms for restricted channel routing," INT. J. Electronics, vol. 77, no. 1, pp. 81-93, 1985.

[8] K. Stecke, "Design, planning, scheduling and control problemsof flexible manufacturing," Annals of Operations Research, vol. 3, pp. 187-209, 1985.

[9] T.F. Coleman and J.J. More, "Estimation of sparse Jacobian Matrices and graph coloring problems," SIAM. J. Numer. Anal., vol. 20, pp 187-209, 1983.

[10] Garey, M.R. and D. S. Johnson, COMPUTERS AND INTRACTABILITY : A GUIDE TO THE THEORY OF NP-COMPLETENESS. Newyork, W. H. Freeman and Co., 1979.

[11] Baase, S. and A.V. Gelder, COMPUTER ALGORITHMS:INTRODUCTION TO DESIGN AND ANALYSIS. Addison-Wesley, 1999.

[12] J.Shawe-Taylor, J.Zerovnik. Generalised Boltzmann Machines, Technical ReportCSD-TR-92-29, Royal Holloway University of London, 1992.

[13] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm", Proceedings of the 1997 Conference on Systems, Man, and Cybernetics, pp. 4104–4109, IEEE Service Center, 1997.

[14] M. Dorigo, V. Maniezzo and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," IEEE Transactions on Systems, Man, and Cybernetics-Part B, vol.26, pp.29-41, 1996.

[15] Noureddine Bouhmala and Ole-Christoffer Granmo, "Solving Graph Coloring Problems using Learning Automata," *European conference on Evoluionary computation in combinatorial optimization,* pp 277-288, Italy,2008.

[16] Tai-wen Yue, Zou Zou Zhonq Lee, "A Q'tron Neural Network Approach to solve the Graph Coloring Problems," *ICTAI, Vol 1, pp 19-23,* 2007.

[17] Pragadeesh Prakasam, Michel Toulouse, Teodor Gabriel crainic, Rong Qu, "Design of a Multilevel Cooperative Heuristic for the Graph Coloring Problem," Interuniversity Research Centre, 2009.

[18] Di Blas, A Jagota, Hughey R, "Energy function –based approaches to graph coloring", *Neural* Networks, Vol. 13, PP. 81-91,IEEE, 2002.

[19] Istvan Juhos, Jano I. van Hemert, "Increasing the efficiency of graph colouring algorithms with a representation based on vector operations",Journal of Software,2006.

[20] Wenbin Liu, Fengyue Zhang and Jin Zu, "A DNA algorithm for the Graph Coloring Problem," ACS Publication, pp. 1176-1178, 2002.

[21] R.Chandresekharam, S.Subhramanian, S.Chaudhury, "Genetic algorithm for node partitioning problem and applications in VLSI design," IEEE, Vol. 140, No. 5, 1993.

[22] Rapha□l Dorne, Jin-Kao Hao, "A New Genetic Local Search Algorithm for Graph coloring," 1998.

[23] Anna Marino, Robert I. Damper, "Breaking the Symmetry of the Graph Colouring problem with Genetic Algorithms," 2000.

[24] Kiyoharu Tagawa, Kenji Kanesige, Katsumi Inoue and Hiromasa Heneda, "Distance Based Hybrid Genetic Algorithm: An application for The Graph Coloring Problem," *IEEE,* 1999.

[25] Cornelius Croitoru, Henri Luchian, Ovidiu Gheorghies, and Adriana Apetrei, "A New Genetic Graph Coloring Heuristic," 2002.

[26] Szymon ukasik, "PARALLEL GENETIC ALGORITHMS FOR GRAPH COLORING PROBLEM USING MESSAGE PASSING PARADIGM," *Journal of ELECTRICAL ENGINEERING*, vol. 56, No. 12, pp. 1-5, 2005.

[27] Yoshiyasu Takefuji and Kuo Chun Lee, "Artificial Neural Networks for Four- Coloring Map Problems and K-Colorability Problems," IEEE transactions on circuits and systems, Vol. 38, No. 3, pp. 326-333, 1991.

[28] Matthias Oliver Berger, "K-Coloring Vertices Using a Neural Network with Convergence to Valid Solutions,"

[29] Arun Jagota, "Approximating Maximum Clique with a Hopfield Network,"

[30] Haidar Harmanani, Jean Hannouche, Nancy Khoury, "A Neural Networks Algorithm for the Minimum Coloring Problem Using FPGAs," IASTED International Conference,Spain, 2003.

[31] S.A.Rahman, Jayadeva and S.C.Dutta Roy, "Neural network approach to graph coloring," Electronics Letters, 1999.

[32] H.J.Kappen and F.B.Rodr guez, "Mean field approach to learning Boltzmann Machines," 1997.

[33] Peter Burge, John Shawe-Taylor, "Bitstream Neurons for Graph Coloring,"

[34] Takashi Yozawa, "A Parallel Algorithm for Boltzmann Machines," pp. 451-460, 1991.

[35] Jieyu Zhao, John Shawe-Taylor, "A Recurrent Network with Stochastic Weights," IEEE, pp. 1302-1307, 1996.

[36] Gustavo Deco, Lucas Parra, "Unsupervised Learning for Boltzmann Machines,"

[37] Peter Burge, John Shawe-Taylor, , "Bitstream Neurons for Graph Colourinng", 1993.

[38] J. H. Holland, "Adpatation in Natural and Artificial Systems", University of Michigan Press, 1975.

[39] C. Reeves (ed.), Modern heuristic techniques for combinatorial problems, Orient Longman, 1993.

[40] S. E.Goodman and S.T.Hedetnieni, *Introduction* to *Design and Analysis of Algorithm*, MGH, 1997.