Enhanced PSO for Graph Coloring Problem

Ankit Singh¹, Dr. Ramveer Singh²

¹M.Tech Student, Computer Science Department, Greater Noida Institute of Technology, Uttar Pradesh, India
² Dr. Ramveer Singh, Information Technology, Greater Noida Institute of Technology, Uttar Pradesh, India

Abstract— In this paper we have proposed a new Particle Swarm Optimization algorithm named Enhanced PSO. The proposed strategy consists of the concept of the Smallest Position Value rule. In this case the solution having the smallest position value will be served first. The newly proposed PSO algorithm provides yet another solution for the planar graph coloring problem using four colors to get smaller average iterations and higher correction coloring rate.

Keywords: Graph Coloring, Graph Coloring Problem, Enhanced PSO, GCP, Modified PSO, SPV.

1. Introduction

Graph coloring is very important part of Graph Theory. It is a special case of graph labeling, which is used to assignment of the labels traditionally called colors to the element of the graph. The Graph Coloring Problem (GCP) is a classical NP-hard problem which has been widely studied [1, 2, 3]. The GCP consist in assigning a color to the vertices of a graph with the limitation that a pair of vertices that are linked cannot have the same color. It can be applied to many engineering applications, such as time tabling and scheduling, radio frequency assignment, computer register allocation, and printed circuit board testing.

The particle swarm optimization (PSO) is a novel multi-agent optimization system (MAOS) inspired by social behavior metaphor developed by Kennedy and Eberhart (1995) [7, 8]. It is one of the algorithm based on the intelligence of the swarm and that increases iteratively. It is a technique used to explore the search space of a given problem to find the settings or parameters required to maximize a particular objective.

In this algorithm each particle maintains its position, composed of the candidate solution and its evaluated fitness, and its velocity. Additionally, it remembers the best fitness value it has achieved thus far during the operation of the algorithm, referred to as the individual best fitness, and the candidate solution that achieved this fitness, referred to as the individual best position or individual best candidate solution. Finally, the PSO algorithm maintains the best fitness value achieved among all particles in the swarm, called the global best fitness, and the candidate solution that achieved this fitness, called the global best position or global best candidate solution. By this method, PSO has been extensively applied for solving optimization problem of continuous space. In 1997, Kennedy proposed a discrete binary PSO algorithm [9, 10]. This version of algorithm not only extends the capabilities of the continuous valued one but also is able to optimize any function, either continuous or discrete.

Ling-Yuan Hsu et al [1] proposed the modified turbulent PSO for solving planar graph coloring problem. In this paper proposed algorithm is combination of the three different strategies. Cui et al [2] proposed the modified PSO for solving planar graph coloring problem. In this paper quaternary PSO algorithm with the disturbance strategy is proposed for improvements in the particles. Jin Qin [3] proposed the hybrid discrete PSO for solving the planar graph coloring problem. Alberto moraglio et al [4] proposed geometric PSO for solving nontrivial combinatorial problems. Sergio Consoli Jose et al [10] proposed a discrete version PSO for solving a graph problem. In this paper, we have proposed a new model for solving the planar graph coloring more quickly. In this paper, improvement of the swarm is obtained from the continuous movement of the particles that constitute the swarm submitted to the effect of the inertia and the attraction of the members who lead the swarm.

The remainder of this paper is organized as follows: Section 2 briefly overviews the procedure of the planar graph coloring. Section 3 describes the particle swarm optimization (PSO) and the new PSO algorithm in detail. Section 4 discusses the experimental results obtained from the new proposed PSO. Finally, Section 5 summarizes the contribution of this paper and concludes.
2. The procedure of the planar graph coloring

In graph theory, a planar graph \([1, 2]\) is a graph that can be embedded in the plane, i.e., it can be drawn on the plane in such a way that its edges intersect only at their endpoints. Many mathematicians had proven that any planar graph could be colored by four kinds of colors, which is called the four-color problem (i.e. four-color conjecture). A coloring of a graph is an assignment of colors to its vertices so that no two adjacent vertices are colored alike. Many scientists have been proven that graph of any kind and adjacency relationship can be simply colored by the four number of colors. So graph coloring problem can be called as four color problem. In mathematics, the four color theorem, or the four color map theorem states that, given any separation of a plane into contiguous regions, producing a figure called a map, no more than four colors are required to color the regions of the map so that no two adjacent regions have the same color.

The four-color problem was originally posed as a conjecture in 1850s. It was finally proved by the american mathematicians Appel and Haken in 1976. Coloring regions (whether these are states, countries, counties) in a map with a minimum number of colors such that neighboring regions (those sharing a common boundary) are colored differently has been proved to be a classic NP-complete problem. In this section, a brief overview of the planar graph coloring is addressed. The procedure of the planar graph coloring is described as follows:

Mathematical description of the planar graph coloring problem: Graph is the combination of the vertices and the edges. Vertices are represented as the set called \(V(g) = \{v_1, v_2, v_3, ..., v_n\}\). And the edges are represented as the adjacency matrix \(r_{ij}\).

\[
\begin{align*}
r_{ij} &= \begin{cases} 
1, & \text{node } i \text{ and } j \text{ connected and } i \neq j; \\
0, & \text{node } i \text{ and } j \text{ unconnected, } \forall i \neq j (1 \leq i, j \leq n)
\end{cases}
\end{align*}
\] (1)

Four different colors are applied to the coloring of all the nodes to make any two adjacent nodes colored in difference. We use 0, 1, 2 and 3 to represent four different colors, and accordingly a reasonable coloring program corresponds to a string with a length of \(n\).

Fig. 1 Four coloring program of seven nodes of graph and its adjacency matrix.

Fig 1 (a) shows the coloring program of the seven node of the given graph.. Fig 1 (b) shows the adjacency matrix of the given graph. Here in this case the set of the vertices are \(V(g) = v_1, v_2, v_3, v_4, v_5, v_6, v_7\). Let the sequence of the colors for the given coloring program is represented as \(S= q_1, q_2, ..., q_n\). Here \(q_i \in \{0,1,2,3\}\). There will also a fitness function that will be taken for the problem environment. That will show the fitness or accuracy of the given solution. We define the fitness value \(f(s)\) as following.

For the node \(v_i \in V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}\) and assigned color to this node is \(q_i\). So for any other node \(v_j \in V\) (where \(i \neq j\) and \(r_{ij} = 1\) ) have same color \(q_i\), will be termed as a conflict. In this case we will create a conflict matrix \(\text{conflict}_{xy}\) that will show the conflict between the two adjacent nodes.

\[
\begin{align*}
\text{conflict}_{xy} &= \begin{cases} 
da_{xy}, & \text{if } r_x = r_y \text{ and } x \neq y \\
0, & \text{otherwise}
\end{cases}
\end{align*}
\] (2)

And fitness value for the given solution will be

\[
f(R) = \sum_{x=1}^{n} \sum_{y=1}^{n} \text{conflict}_{xy},
\] (3)
3. Enhanced PSO

PSO [7, 8] is an efficient, robust and simple optimization algorithm for solving many continuous optimization problems. It is inspired by the social behavior of the swarm of the birds. This is the specialized concept of the evolutionary computation. In particle swarm optimization, each potential solution is assigned with the randomized velocity and the position. Here potential solution is called the particle and sometimes also called the candidate solution. The particle will move through the problem space unless particle reach to the global best position or any one particle reach to the best or optimum solution.

3.1. Standard particle swarm optimization

In computer science, particle swarm optimization optimizes problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. PSO produce the optimum result of the problem by having the $n$ number of the candidate solutions of the particular problem and the movement of the solutions or the particle on the basis of its position and the current velocity as well as on the basis of the global best particle. We can simply say that movement of the particle will be influenced by the global best position and the local best position. It was developed by the Kennedy and the Eberhart. The concept of the PSO was taken from the concept of the movement of the birds flock or the fish school. A basic variant of the PSO algorithm works by having a population (called a swarm) of candidate solutions (called particles). These particles moved around in the search-space according to a few simple formulae. The movements of the particles are guided by their own best known position in the search-space as well as the entire swarm’s best known position.

In the particle swarm optimization, each particle $i$ has a position represented by a position vector $P_i$. A swarm of particles moves through a multi-dimensional problem space and each particle $i$ with the velocity represented by a vector $V_i$. Each particle keeps track of its own best position in each iteration (or time cycle), which is associated with the best experience it has achieved and denotes $P_{best}$. The best position among all the particles obtained in the population denotes $P_{Gbest}$. For each iteration, the position $P_{best}$ of its own best and the position $P_{Gbest}$ of the best particle of swarm are calculated as the best fitness of all particles. Accordingly, each particle changing the velocity this way enables the particle $P_i$ to search around its individual best position $P_{best}$ and the global best position $P_{Gbest}$ as follows:

$$V_i = w \times V_i + c_1 \times \text{rand}(i) \times (P_{\text{best}} - P_i) + c_2 \times \text{rand}(i) \times (P_{\text{Gbest}} - P_i) \quad (4)$$

And the new position will be

$$P_i = P_i + V_i \quad (5)$$

The whole running procedure of the PSO algorithm is as follows: *(Standard PSO algorithm)*

1. Initialize all particles’ positions $P_i$ and velocities $V_i$, for $1 \leq i \leq \text{NumberOfParticles}$.
2. while the stop condition (the optimal solution is found or the maximal moving steps are reached) is not satisfied do
3. for particle $i$, $(1 \leq i \leq \text{NumberOfParticles})$ do
4. Calculate the fitness value of particle $i$.
5. Update the personal best position of the particle $i$ according to the fitness value.
6. End for
7. Update the global best position of the particles according to the fitness value.
8. For Particle $i$, $(1 \leq i \leq \text{NumberOfParticles})$ do
9. Move particle $i$ to another position according to the equation (4) and (5).
10. End for
11. End while

3.2. Discrete particle swarm optimization

The original PSO algorithm can only optimize problems in which the elements of the solution are continuous real numbers since, in words of the inventors of PSO, it is not possible to “throw to fly” particles in a discrete space. In the last years, several modifications of the PSO algorithm for solving problems with discrete variables have been proposed in the literature. They are referred to as Discrete Particle Swarm Optimization (DPSO) methods [9, 10]. Kennedy and Eberhart developed a DPSO algorithm for problems with binary-valued solution elements where the position of each particle is a vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ of the d-dimensional binary solution space, $x_i \in \{0, 1\}^d$, but the velocity is still a vector $v_i = (v_{i1}, v_{i2}, \ldots, v_{id})$ of the d-dimensional continuous space. As with the standard PSO strategy by Kennedy and Eberhart, the velocity is still updated by means of Eq.4. However the significance of the velocity term has been changed to indicate the probability of the corresponding solution element assuming a value of 0 or 1. In other words, the continuous value $v_{ij}$ refers to
the probability that the j-th binary variable within the position of the i-th particle, \( x_{ij} \), assumes a value of 0 or 1 at the next iteration. For assigning a new position value to a particle i, each position variable \( x_{ij} \) is randomly set with probability of selecting a value of 1 given by the sigmoid function:

\[
\frac{1}{1 + \exp(-v_{ij})}
\]

Cui et al. [2] developed a quaternary-valued PSO method by defining the particles’ positions and velocities in terms of changes of probabilities of solution elements’ assumed values. Thus a particle moves in a state space restricted to zero, one, two, and three on each dimension, where each particle’s velocity represents the probability of particle’s position taking the assumed value. For assigning a new particle position, the quaternary-valued PSO method uses Eqs. (6) and (7) to replace Eq. (5).

\[
P_j = \text{Mod}(P_j + f(V_j)), 4),
\]

\[
f(V_j) = \begin{cases} 
0, \text{rand}() > r \& \text{rand}() < S(V_j), \\
1, \text{rand}() < r \& \text{rand}() < S(V_j), \\
2, \text{rand}() \geq r \& \text{rand}() \geq S(V_j), \\
3, \text{rand}() < r \& \text{rand}() \geq S(V_j), 
\end{cases}
\]

3.3 New Planar Graph Coloring Model

A new planar graph coloring model named Enhanced PSO based on the concept of the Smallest Position Value. This model is inspired by the literature [1]. This model employs the walking one strategy, greedy algorithm, swapping strategy and SPV rule.

3.3.1 Smallest Position value

SPV rule [5] is used for changing the continuous value in discrete value. SPV rule states that vertex with the smallest position value is scheduled first and vertex with second SPV comes next. We represent the dimension of the swarm as the number of vertices. Position vector.

\[ P_{id} = \{p_1, p_2, p_3, \ldots, p_d\} \]

Where \( i \) is the particular individual vertex and \( d \) represents the dimension index, is calculated using PSO. The position vector of each particle makes transformation about the continuous position. Smallest position value rule is used to find a permutation corresponding position \( P_{id} \). The position vector \( P_{id} \) has continuous values. By using the SPV rule this continuous position value can be converted to discrete value permutation

\[ Q_{id} = [q_{i1}, q_{i2}, \ldots, q_{id}] \]

where \( Q_{id} \) is the sequence of nodes of \( i \) particle in the processing order with respect to the \( d \) dimension. Set of the colors is represented by

\[ C_{lm} = \{r_1, r_2, r_3, \ldots, r_y\} \]

Where \( l \) represents the particle and \( m \) represent the color with which vertex is colored. After SPV, set of colors is calculated using equation

\[ C_{im} = Q_{id} \text{ mod } M \]

3.3.2 Walking one strategy

Cui et al. developed a quaternary-valued PSO method by defining the particles positions and velocities. The walking one strategy is a probability function based on quaternary valued PSO; it depends on the node and adjacent nodes of the number of the conflicts. The number of the conflict nodes could get from Eq. (8) based on Eq. (2). Then the number of the conflict nodes will be converted to collision factor through the sigmoid function (i.e. Eq. (9)). If the collision factor is large, then it will be assigned a higher probability to change its color number. Otherwise, it will be assigned a lower probability. (i.e. Eqs. (10) and (11)). The walking one strategy is shown as follows:

\[ C_{r_j} = \sum_{k=1}^{n} \text{conflict}_k, \quad (8) \]

\[ C_{f_j} = \frac{1}{1 + e^{-C_{r_j}+2}}, \quad (9) \]
\[ M(V_j) = \begin{cases} 1, & \text{if } C_f > \text{rand()} \& S(V_j) > \text{rand()} \\ 0, & \text{otherwise} \end{cases} \]  

\[ P_j = \text{Mod}((P_j + M(V_j)), 4) \]  

where \( C_r(1 \leq j \leq n) \) is the number of conflict nodes with \( j \)th node, and \( C_f \) is the collision factor of \( j \)th node in the range \([0, 1]\), \( n \) is the number of nodes, \( \text{rand()} \) is a uniformly distributed random number in the range \([0, 1]\), \( S(V_j) \) is the sigmoid function given by \( S(V_j) = \frac{1}{1 + e^{-v}} \).

### 3.3.3 Greedy algorithm

Greedy strategy deals with the maximum conflicting node first and assess it. Because in graph coloring maximum conflicting node is most troublesome and need to be processed first.

We will first create the conflict array \( C_r \) of all nodes of the graph that will contain the number of the conflicts for all nodes of the graph. The algorithm for serving the maximum conflicting node will be as follows.

**Algorithm 1**

1. Sort the \( C_r \) in descending order pick the first element of the \( C_r \) (\( C_{r1} \)). Check the label (\( l \)) of the max conflicting node in the graph.
2. Counter = false
3. Calculate the fitness value of the particle \( f_{old} \).
4. List all the number of the colors used by the neighbor of the maximum conflicting node.
5. If (all available colors are utilized by the neighbors)
   6. Counter = true
   7. End if
8. Else (Find the color ‘\( o \)’ which is not in the list of the colors of the neighbors.)
   9. \( P_l = o \)
10. End else
11. Calculate finess value \( f_{new} \) of particle \( l \)
12. If(\( f_{new} < f_{old} \))
13. Accept color \( o \)
14. End if

### 3.3.4 Swapping Strategy

This strategy is combination of the two strategy swapping and the inversion. In swapping strategy node have maximum number of adjacency will be swapped with last node of the graph in sequence. While in inversion strategy color sequence of nodes after the node having maximum number of adjacency will be inversed. Both can be easily understand by the given color sequence.

![Fig. 2 swapping strategy](image)

![Fig. 3 inversion strategy](image)

### 3.4.4 Enhanced PSO (Algorithm 2)

1. Initialize all the particle’s position(i.e. coloring number) as per the SPV rule and velocities \( V_i \) for \( (1 \leq i \leq \text{NumberOfColors}) \)
2. While the stop condition (optimal solution is found or maximum moving steps are reached) do
3. For particle \( (1 \leq i \leq \text{NumberOfColors}) \) do
4. Update personal best position of the particle and also update global best position of the particle
5. End for
6. For particle \((1 \leq i \leq \text{NumberOfColors}) \)
7. Calculate the velocity of the particle by the standard equation no. (4) of the PSO.
8. Change the position(coloring sequence) of the particle by the algorithm 1
9. If counter = true then
10. Calculate the \( C_{r1} \) by equation (8)
11. Calculate \( C_{r0} \) by equation (9)
12. Calculate \( M(V_j) \) by equation (10)
13. Calculate \( P_j \) by equation (11)
14. Evaluate the quality of the fitness \( F_j \)
15. Swap node
16. Evaluate the quality of the fitness \( F_j \)
17. Inverse node
18. End if
19. End for
20. End while
4. EXPERIMENTAL RESULTS

In this section the performance of the Enhanced PSO algorithm is investigated by some experiments. All computation experiments are conducted with the visual studio, and run on Intel® Core(TM)2 Duo CPU P9400 @2.40 GHz with 2 GB memory capacity. The essential parameter for Enhanced PSO algorithm is set as follow. We simulated 30 runs. Let maximum number of evaluations be 20000 and maximum swarm size is 20.

In order to validate the results a comparison of the performance of the Enhanced PSO algorithm with MPSO algorithm (the number of particles as 20) is presented. By randomly generating a given scaled planar graph and calculating 30 runs of the Enhanced PSO and MPSO, the experimental results we obtained are shown in Table 1. In order to demonstrate the performance of Enhanced PSO algorithm for solving the planar graph coloring problem further, we take the coloring problem of the random Map as an example. Fig.4 is the randomly taken graph, including 30 vertices. We respectively used the MPSO and Enhanced PSO to perform the experiment. The results demonstrate that the Enhanced PSO method is able to present a feasible coloring scheme to this problem.

<table>
<thead>
<tr>
<th>Node</th>
<th>Algorithm</th>
<th>Average iterations</th>
<th>Average evaluations</th>
<th>Success rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>MPSO</td>
<td>7.40</td>
<td>129</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Enhanced PSO</td>
<td>5.00</td>
<td>90</td>
<td>100%</td>
</tr>
<tr>
<td>10</td>
<td>MPSO</td>
<td>37.20</td>
<td>614.40</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Enhanced PSO</td>
<td>7.00</td>
<td>128</td>
<td>100%</td>
</tr>
<tr>
<td>20</td>
<td>MPSO</td>
<td>139.96</td>
<td>2540.40</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Enhanced PSO</td>
<td>13</td>
<td>252</td>
<td>100%</td>
</tr>
<tr>
<td>25</td>
<td>MPSO</td>
<td>3498.60</td>
<td>70000</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>Enhanced PSO</td>
<td>24.40</td>
<td>508.00</td>
<td>100%</td>
</tr>
</tbody>
</table>

The best individual color coding is 2 3 0 2 1 0 1 2 0 1 2 3 3 0 3 1 0 1 3 2 2 0 3 2 0 1 1 0.

5. CONCLUSION

In this paper, we have proposed an Enhanced particle swarm optimization with smallest position value for solving planar graph coloring. In this scheme we are using the concept of walking one strategy and greedy algorithm for improving the discrete PSO in the planar graph coloring. The walking one strategy and greedy strategy has improved the PSO algorithm for solving Graph Coloring Problem. The inversion strategy can be helpful to drive particles (i.e., those particles that are much far from optimum solution) and hence they can explore a better solution. The experimental results show that this algorithm is considerably effective to the graph coloring problem with moderate size and provides another solution for planar graph coloring problem.
REFERENCES


[3]. Jin Qin (2011), Hybrid Discrete Particle Swarm Algorithm for Graph Coloring Problem, Journal of computers volume 6, number 6, pages 1175-1182.

[4]. Alberto moraglio, Julian Togelius (2007), Geometric Particle Swarm Optimization for the Sudoku Puzzle, ACM-USA.


[10]. Sergio Consoli Jose, Andre's Moreno-Perez, Kenneth Darby-Dowman, Nenad Mladenovic (2009), Discrete Particle Swarm Optimization for the minimum labeling Steiner tree problem, Springer science.

BIOGRAPHIES

Ankit Singh, Pursuing his Master degree in Computer Engineering from GNIOT, Gr Noida. His area of interest is Algorithm Design & Analysis and Graph Theory.

Dr. Ramveer Singh, HoD and Professor, Information Technology, GNIOT, Gr Noida. His area of interest is Network Security And Cryptography.