PSO Algorithm with Transition Probability Based on Hamming Distance for Graph Coloring Problem

TAKUYA AOKI
Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba, Japan

CLAUS ARANHA, HITOSHI KANOH
Division of Information Engineering, Faculty of Engineering, Information and Systems, University of Tsukuba, Japan

Abstract—In this paper, we propose a PSO algorithm with transition probability based on Hamming distance for solving planar graph coloring problems. PSO was originally intended to handle only continuous optimization problems. To apply PSO to discrete problems, the standard arithmetic operators of PSO need to be redefined over discrete space. In this work, we propose a new algorithm that uses transition probability based on Hamming distance into PSO. The experimental results show that the new algorithm can get higher success rate and smaller average iterations than a Genetic Algorithm and the conventional PSO.

Keywords—Particle Swarm Optimization; Graph Coloring Problem; Hamming distance; discrete PSO

I. INTRODUCTION

Particle Swarm Optimization (PSO) is a population-based stochastic search algorithm that is inspired by the social interaction behavior of birds and proposed by Kennedy and Eberhart in 1995 [1]. The first version of PSO was intended to solve only continuous optimization problems. As many optimization problems are defined in the discrete space, many works tried to extend the PSO to solve discrete optimization problems [2]. A typical algorithm is Binary PSO (BPSO) [3]. BPSO extended the PSO to solve binary optimization problems by using the sigmoid function (SF).

BPSO has itself been extended in many ways, solving other discrete optimization problems, such as the knapsack problem [4][5][6]. Another approach is to use the smallest position value instead of SF. This approach was used to solve the shop sequencing problem, and the travelling salesman problem [7][8]. Several extended BPSO have been proposed to solve the graph coloring problem (GCP) [9][10].

Chen et al. introduced transition probability into PSO (TPPSO) to solve the timetabling problem [11]. However, this method does not consider the distance between solutions. This model does not seem to search effectively, because the searching always performs in the same way whether the particle is close the optimal solution or not. In this paper, we improve Chen’s method and change the transition probability so that it is based on the Hamming distance.

Although various PSO algorithms have been proposed for discrete optimization problems, they generally do not show a greater performance when compared with the other metaheuristics algorithms [12]. On the other hand, the hybrid PSO algorithms’ performance are greater than the pure PSO approaches. However, as these hybrid algorithms are generally designed for specific problems, their structures are more complicated, and they are harder to generalize [12]. The proposed algorithm not only has higher performance on the GCP, it is also flexible to other problems.

In the following sections, we first discuss the original PSO algorithm, the GCP and related works. Next, we describe the proposed method in detail. Finally, we give the results of experiments solving random generated GCP and show that this approach is a more effective solution than TPPSO and Genetic Algorithm (GA).

II. PROBLEM DESCRIPTION

A. Original PSO

Particle swarm optimization (PSO) is a stochastic search algorithm for problem solving that is inspired by simulations of the swarm behavior of birds flocking. In PSO, a number of individuals, called particles, are placed in the search space of a problem. The each particle is a point in N-dimensional search space. Each particle evaluates its fitness, i.e. the objective function, at its current location. The coordinates of each particle represent a possible solution associated with two vectors: the position and the velocity vectors.

Each particle adjusts its flying in accordance with its own and its companions’ flying experience. The position and velocity of i-th particle are represented as \( x_i = (x_{i1}, x_{i2}, \ldots, x_{iN}) \) and \( v_i = (v_{i1}, v_{i2}, \ldots, v_{iN}) \), respectively.

The best previous position, i.e. the position giving the best fitness value, for particle \( i \) is represented as \( pbest_i = (pbest_{i1}, pbest_{i2}, \ldots, pbest_{iN}) \) and the best previous position among all particles in the population is represented as \( gbest = (gbest_1, gbest_2, \ldots, gbest_N) \). Each particle updates its position in accordance with (1) and (2), where \( k \) is the iteration index, \( c_1 \) and \( c_2 \) are constants, \( r_1 \) and \( r_2 \) are uniform random numbers in the range of \([0,1]\), and \( w \) is the inertia weight. The next iteration takes place after all particles have been moved. Eventually the swarm as a whole, like a flock of birds collectively foraging for food, is likely to move close to an optimum of the fitness function.

\[
v_{i}^{k+1} = wv_{i}^{k} + c_1r_1(pbest_{i}^{k} - x_{i}) + c_2r_2(gbest^{k} - x_{i}) . \tag{1}
\]

\[
x_{i}^{k+1} = x_{i}^{k} + v_{i}^{k+1} . \tag{2}
\]
B. Graph Coloring Problem

A graph 3-coloring problem consists of an undirected graph with \( n \) vertices. Each vertex must be assigned one of three colors subject to the constraint that no neighboring vertex is assigned the same color. As an example, see Fig.2, where R is red, G is green, B is blue. This problem is a well-studied NP-complete problem that is used to model certain types of scheduling and resource allocation problems, such as examination scheduling.

Solvable problems with \( n \) nodes and \( m \) arcs are generated as follows\([13]\):

1. Create three groups of nodes, each with \( \frac{n}{3} \) nodes.
2. Randomly create \( m \) arcs between nodes in different groups.
3. Accept the graph if it has no unconnected components.

And we define a constraint density \( d = \frac{m}{n} \). The constraint density is an index to grouping problems by its level of difficulty. Fig. 1 shows the image of the relationship between constraint density and difficulty. According to Hogg’s research \([14]\), most difficult problem is \( d = 2.0-2.5 \). This is because the loose constraints allow for many local optimal solutions.

![Figure 1. Image of the relationship between constraint density and difficulty](image)

One example of GCP can be seen on Fig. 2, along with a possible solution. We express a candidate solution for GCP as the coloring sequence \( x_i = (x_{i1}, x_{i2}, \ldots, x_{in}) \), where \( x_{ij} \in \{0, 1, 2\} \). Numbers 0, 1, and 2 correspond to R, G and B respectively. According to this coding, the solution in Fig. 2 can be expression as \( x_i = (0, 0, 1, 2) \).

![Figure 2. An example of GCP](image)

C. Related Works

1) MTPSO: Hsu et al. proposed a modified turbulent particle swarm optimization (MTPSO) model based on PSO for solving graph coloring problems. The proposed model consists of the walking one strategy, assessment strategy and turbulent strategy. The proposed MTPSO model can solve the graph coloring problem using four-colors efficiently and accurately.

Compared to the results of pure PSO and Modified PSO shown in Cui et. al. (2008), not only the experimental results of the proposed model gets smaller average iterations but also higher correction coloring rate when the number of nodes is greater than 30.

However, in this method, the definition of distance has some problems. For example, the distance between red to green is 1, and the one between red to blue is 2, but these two distances are essentially the same.

2) Transition Probability PSO: Although the mainstream way of modifying PSO for discrete problems is by using SF, Chen solved university course timetabling problems using PSO with transition probability \([11]\). Experiments using timetables of the University of Tsukuba showed that this approach is an effective solution. The proposed method includes a general technique that can be applied to various large-scale real-life Combinatorial Optimization Problems (COPs) and Constraint Satisfaction Problems (CSPs). So it can be applied to GCP.

Transition probability to each color is the probability that a vertex a particle will change to that color. This probability consists of three connected probabilities: random color \( P_{\text{rand}} \), a color of the vertex on the personal best particle \( P_{\text{pbest}} \), and a color of the vertex on the global best particle \( P_{\text{gbest}} \).

Each probability can be defined by (3)-(5), where \( w, c_1 \), and \( c_2 \) are constants defined by experiments, \( r_1 \) and \( r_2 \) are uniform random numbers in the range of \([0, 1]\).

\[
P_{\text{rand}} = \frac{w}{w + c_1 r_1 + c_2 r_2} \quad (3)
\]

\[
P_{\text{pbest}} = \frac{c_1 r_1}{w + c_1 r_1 + c_2 r_2} \quad (4)
\]

\[
P_{\text{gbest}} = \frac{c_2 r_2}{w + c_1 r_1 + c_2 r_2} \quad (5)
\]

The calculation of the transition probability in the proposed method corresponds to the calculation of velocity in continuous PSOs. This method can decrease influence of encoding colors as numbers, but does not consider the distance among particles. As a result, even if a particle is closer to \( \text{pbest} \) or \( \text{gbest} \), the searching always performs in the same way.

III. PROPOSED METHOD

A. Basic strategies

To make good use of transition probability and consider the distance among particles, we changed the definition and operation of the transition probability.

First, to compare the positions of particles, we introduce the Hamming distance as the distance among particles.

Next, to take into account the internal relationship between colors in one particle, we compute transition probabilities for each particle, instead of each color. This makes computational cost dependent on the number of particles, not colors. In
addition, it improves the search diversity, because each particle has multiple transition probabilities.

Finally, we changed the operation of transition probability. The basic idea is that the smaller the Hamming distance between a particle and \( pbest_i \) or \( gbest_i \), the bigger the transition probability changing to \( pbest_i \) or \( gbest_i \) of the particle will be. To avoid being trapped in local best, the smaller the Hamming distance between the present position of a particle and the previous position of it is, the more the particle moves randomly.

B. Transition Probability

We proposed a new transition probability based on the Hamming distance. This probability consists of three connected probabilities: a random color \( P_{rand} \) a color of the vertex on the personal best particle \( P_{pbest} \), and a color of the vertex on the global best particle \( P_{gbest} \); see (15) and procedure update-position. The probabilities can be defined by (7) to (14). And we defined similarity by (6), where \( H(x,y) \) is Hamming distance between \( x \) and \( y \).

\[
x \oplus y = 1 - \frac{H(x,y)}{n} \quad (6)
\]

\[
v_i^k = (x_i^k \oplus x_i^{k-1}) \quad (7)
\]

\[
V_{rand} = wv_i^k \quad (8)
\]

\[
V_{pbest} = c_1r_1(x_i^k \ominus pbest_i^k) \quad (9)
\]

\[
V_{gbest} = c_2r_2(x_i^k \ominus gbest_i^k) \quad (10)
\]

\[
V = V_{rand} + V_{pbest} + V_{gbest} \quad (11)
\]

\[
P_{rand} = \frac{V_{rand}}{V} \quad (12)
\]

\[
P_{pbest} = \frac{V_{pbest}}{V} \quad (13)
\]

\[
P_{gbest} = \frac{V_{gbest}}{V} \quad (14)
\]

\[
x_{ij}^k = \begin{cases} 
\text{random color} & P_{rand} \\
pbest_i^k & P_{pbest} \\
gbest_i^k & P_{gbest} 
\end{cases} \quad (15)
\]

C. Objective Function

The objective function of each particle in the population is measured by the degree to which the particle meets the constraints. The objective function value (fitness) of the \( i \)-th particle \( f_i \) can be expressed by (16), where conflict(\( x \)) is the number of conflicts of \( x \). A conflict happens when two linked vertices are assigned the same color.

\[
f_i = 1 - (\text{conflict}(x_i) / m) \quad (16)
\]

D. Pseudo Code

The main procedure and the procedure for one update of position in the flying process of \( i \)-th particle \( x_i \) are as follows.

**Procedure main()**

- Initialize all particles;
- Evaluate fitness of all particles;
- Save gbest and all pbests;
- While (the stop condition is not satisfied) {
  - for (\( i = 1 \) to Population size) {
    - Calculate \( P_{rand}, P_{pbest}, \) and \( P_{gbest} \);
    - update-position(\( x_i \));
    - Evaluate fitness of particle \( i \);
    - Update pbest;
  }
  - Update gbest;
}

**Procedure update-position(\( x_i \))**

- for (\( j = 1 \) to \( N \)) {
  - Generate number \( r \in [0, 1] \);
  - if \( r \leq P_{rand} \) \{ \( x_{ij} = \) random color; \}
  - else if \( \text{rand} \leq P_{rand} + P_{pbest} \) \{ \( x_{ij} = pbest_{ij} \); \}
  - else \{ \( x_{ij} = gbest_{ij} \); \}
}

IV. EXPERIMENTS

A. Parameter Dependence

We first examined the dependence of the success rate on the parameters \( w, c_1, \) and \( c_2 \) through experiments, where the node size was 90, the constraint density was 2, and other parameters are shown in TABLE 1, where \( E_r \) is the best value of parameter \( x \). A hyphen in the table indicates the parameter under study in the experiment.

Figs. 3 to 5 show the experimental results. Each point in the Figures is the average of 100 trials using different random generated graphs. The optimal values of the parameters obtained by the experiments were \( w = 0.05, c_1 = 7.0, \) and \( c_2 = 0.03 \).

<table>
<thead>
<tr>
<th>TABLE 1. Experimental conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp. 1</td>
</tr>
<tr>
<td>Population size</td>
</tr>
<tr>
<td>Max iteration</td>
</tr>
<tr>
<td>(w)</td>
</tr>
<tr>
<td>(c_1)</td>
</tr>
<tr>
<td>(c_2)</td>
</tr>
<tr>
<td>Result</td>
</tr>
</tbody>
</table>
Second, we examined searching progress of the average of 100 trials, where the node size was 90, the constraint density was 2, and other parameters are optimized. Figure 13 shows the experimental results. We can see the convergence of fitness in Fig. 6.

B. Comparative Study

To evaluate the performance of the proposed method, we compared it with GA and with TPPSO [11]. The parameters are shown in TABLE 2. The parameters for the TPPSO are taken from Chen’s research. The parameters of GA are optimized using a setup similar to section A. The experimental results we obtained are shown in Figs. 7 to 12.

<table>
<thead>
<tr>
<th>Proposed</th>
<th>TPPSO</th>
<th>GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Population size</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Max iteration</td>
<td>$1.0 \times 10^7$</td>
<td>$1.0 \times 10^5$</td>
</tr>
<tr>
<td>$w$</td>
<td>0.05</td>
<td>0.02</td>
</tr>
<tr>
<td>$c_1$</td>
<td>7.0</td>
<td>2.5</td>
</tr>
<tr>
<td>$c_2$</td>
<td>0.03</td>
<td>1.0</td>
</tr>
<tr>
<td>Mutation rate</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Tournament size</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Elite size</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Figure 3. Relationship between success rate and $w$

Figure 4. Relationship between success rate and $c_2$

Figure 5. Relationship between success rate and $c_1$
1) **Success Rate**: From Figs. 7 to 9, we can see the following:
   - For graphs of $n = 90$ and $d = 2.0$ to $5.0$, the proposed method shows higher success rate than others.
   - For graphs of $n = 120$ and $d = 1.5$ to $4.0$, the proposed method shows higher success rate than others.
   - For graphs of $n = 150$ and $d = 1.5$ to $6.0$, the proposed method shows higher success rate than others.
   - For graphs of $n = \{90, 120, 150\}$ and $d = 7.0$ to $10.0$, all methods’ success rates are almost 100%.
   - For graphs of $n = \{90, 120, 150\}$ and $d = 2.5$, which are the most difficult problems, the proposed method shows higher success rate than others.

2) **Iteration Average**: From Figs. 10 to 12, we can see the following:
   - For graphs of $n = \{90, 120, 150\}$ and $d = 1.5$ to $4.0$, the proposed method shows lower average iterations than GA.
   - For graphs of $n = \{90, 120, 150\}$ and $d = 6.0$ to $10.0$, the proposed method shows lower average iterations than others. Because no solution can be found at $d=2.5$, the line is indicated by dotted line.
the GA shows lower average iterations than the proposed method.

- For graphs of \( n = 90 \) and \( d = 6.0 \) to 10.0, the TPPSO shows lower average iterations than the proposed one.
- For graphs of \( n = 120 \) and \( d = 2.5 \) to 3.0 and 5.0 to 10.0, the TPPSO shows lower average iterations than the proposed one.
- For graphs of \( n = 150 \) and \( d = 2.0, 3.0 \) and 6.0 to 10.0, the TPPSO shows lower average iterations than the proposed one.

The above results suggest that proposed method is effective.

C. Discussion

Although we obtained the optimized parameters in Figs. 3 to 5, it seems that the parameter \( c_2 \) is much smaller than \( c_1 \). We examined the dependence of the parameters on constraint density through experiments, where the node size was 90. Fig. 13 shows the experimental results.

The best value for \( w \) was about 0.05. It had small variations according to the value of \( d \), but these variations did not show a statistically significant difference on a Wald proportion test. The optimal values of \( c_1 \) and \( c_2 \) have a cross point at about 6.5. This means that \( p_{best} \) and \( g_{best} \) are more effective in searching when \( d < 6.5 \) and \( d > 6.5 \), respectively.

![Figure 13. Dependence of the parameters on constraint density](image)

V. Conclusions

In this paper, we solved graph coloring problems using PSO with transition probability based on Hamming distance. Experiments using random generated graphs showed that this approach is an effective solution. And we found the cross point of the parameters’ effectivity. In future work, we will research why does the cross point arise and try to apply our method to a real world application, such as scheduling problems.

VI. References