Expert Systems with Applications 38 (2011) 5525–5531

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

MTPSO algorithm for solving planar graph coloring problem ☆

Ling-Yuan Hsu [a,1], Shi-Jinn Horng [a,b,*], Pingzhi Fan [b], Muhammad Khurram Khan [c], Yuh-Rau Wang [d], Ray-Shine Run [e], Jui-Lin Lai [e], Rong-Jian Chen [e]

[a] Department of Computer Science & Information Engineering, National Taiwan University of Science and Technology, 106 Taipei, Taiwan
[b] Institute of Mobile Communications, Southwest Jiaotong University, Chengdu, Sichuan 610031, PR China
[c] Center of Excellence in Information Assurance, King Saud University, Saudi Arabia
[d] Department of Computer Science and Information Engineering, St. John's University, Taipei, Taiwan
[e] Department of Electronic Engineering, National United University, 36003 Miao-Li, Taiwan

ARTICLE INFO

Keywords:
Planar graph coloring
Four-color problem
Modified turbulent particle swarm optimization
Walking one
PSO

ABSTRACT

In this paper, we proposed a modified turbulent particle swarm optimization (named MTPSO) model for solving planar graph coloring problem based on particle swarm optimization. The proposed model is consisting of the walking one strategy, assessment strategy and turbulent strategy. The proposed MTPSO model can solve the planar graph coloring problem using four-colors more efficiently and accurately. Compared to the results shown in Cui et al. (2008), not only the experimental results of the proposed model can get smaller average iterations but can get higher correction coloring rate when the number of nodes is greater than 30.

1. Introduction

The graph coloring problem has been proved to be a classic NP-complete problem. Until now, there is not an effective strategy to get the best solution. For solving this kind of problem, both the exact algorithms and approximate algorithms have been used including ant colony optimization algorithm (Bui, Nguyen, Patel, & Phan, 2008; Dowsland & Thompson, 2008; Salari & Eshghi, 2005; SangHyuck, SeungGwan, & TaeChoong, 2003), tabu search algorithm (Blochliger & Zufferey, 2008; Galinier & Hertz, 2006; Osman, 2006), genetic algorithm (Yanez & Ramirez, 2003), particle swarm optimization algorithm (Cui et al., 2008), neural network algorithm (Talavan & Yanez, 2008), etc. It can be applied to many engineering applications, such as time tabling and scheduling (de Werra, 1985; Leighton, 1979), radio frequency assignment (Gamst, 1986), computer register allocation (Chaitin et al., 1981; Chow & Hennessy, 1990), and printed circuit board testing (Garey, Johnson, & So, 1976).

The particle swarm optimization (PSO) is a novel multi-agent optimization system (MAOS) inspired by social behavior metaphor developed by Kennedy and Eberhart (1995, 1995). It has been widely applied to solve various optimization problems, such as Kuo et al. applied the PSO and fuzzy time series to forecast enrollments (Kuo et al., 2009), Kuo et al. invoked the hybrid PSO for flow-shop scheduling (Kuo et al., 2009), Park et al. applied two-factors high-order fuzzy time series and PSO to forecast TAIFEX and KOSPI 200 (Park, Lee, Song, & Chun, 2010). It is unlike an evolutionary algorithm, however, in that each potential solution is also assigned a randomized velocity, and the potential solutions, called particles, are then "flown" through the multi-dimensional problem space. By this method, PSO has been extensively applied for solving optimization problem of continuous space. In 1997, Kennedy proposed a discrete binary PSO algorithm (Kennedy & Eberhart, 1997). 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.

By solving planar graph coloring problem using PSO, Cui et al. (2008) proposed the modified PSO, which is superior to that of the classical PSO. In this paper, we propose a new model, which uses "modified turbulent particle swarm optimization" (named MTPSO) techniques for solving the planar graph coloring more quickly. The experimental results show that the new model is more efficient than the modified PSO proposed by Cui et al. (2008).

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). Section 4 discusses the new proposed planar graph coloring

☆ This work was supported in part by the National Science Council under contract number NSC-98-2219-E-011-002, NSC-98-2221-E-011-133-MY3, NSC-98-2923-E-011-004-MY3, and NSC-99-2916-I-011-002-A1.

* Corresponding author at: Department of Computer Science & Information Engineering, National Taiwan University of Science and Technology, 106 Taipei, Taiwan. Tel.: +886 2 27376700; fax: +886 2 27301081.

E-mail addresses: D9415008@mail.ntust.edu.tw (L.-Y. Hsu), horngsj@yahoo.com.tw (S.-J. Horng), p.fan@ieee.org (P. Fan), mkhurram@ksu.edu.sa (M.K. Khan), yrwang@mail.sju.edu.tw (Y.-R. Wang), run5116@ms16.hinet.net (R.-S. Run), jllai@nuu.edu.tw (J.-L. Lai), rjchen@nuu.edu.tw (R.-J. Chen).

[1] Present address: Department of Information Management, St. Mary's Medicine Nursing and Management College, I-Lan, Taiwan.

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.10.084

algorithm in detail. Section 5 discusses the experimental results obtained from the new proposed planar graph coloring model. Finally, Section 6 summarizes the contribution of this paper and conclusions.

2. The procedure of the planar graph coloring

What is the minimum number of colors that can be used to color the regions in a planar map with neighboring regions having different colors? This has been a problem of interest for over a century. As early as 100 years ago there were many scholars who had been attracted to carry on researches on this problem, and 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). 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:

Step 1: Transferring the map to a graph

It is not particularly difficult to show that the map can be colored with four-colors, shown in Fig. 1, that is, each region of the map can be assigned one of four given colors such that neighboring regions are colored differently. So, with each map, there is associated a graph G, called the dual of the map, shown in Fig. 2, whose vertices are the regions of the map and such that two vertices of G are adjacent if the corresponding regions are neighboring regions. Observe that the graph G of Fig. 2 is a connected planar graph, shown in Fig. 3.

Step 2: Creating the adjacency matrix of graph

As we know, a graph G can be defined by two sets, namely its vertex set $V(G)$ and edge set $E(G)$ as described in Eqs. (1) and (2), respectively.

$$V(G) = \{v_1, v_2, v_3, \ldots, v_n\}, \tag{1}$$

$$E(G) = \{e_1, e_2, e_3, \ldots, e_m\}, \tag{2}$$

where n is the number of nodes and m is the number of edges. A graph can also be described by an adjacency matrix using Eq. (3). For example, the adjacency matrix A for Fig. 3 is shown in Fig. 4.

$$a_{ij} = \begin{cases} 1 & \text{if } v_i v_j \in E(G), \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

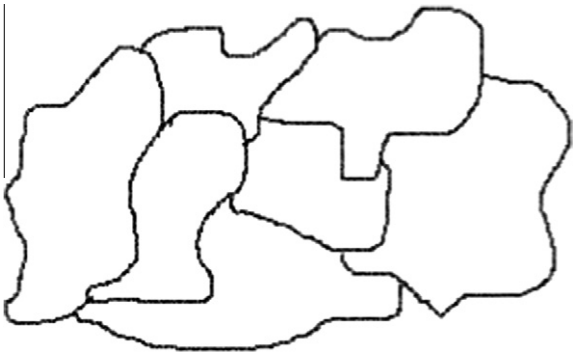

Fig. 1. The original map.

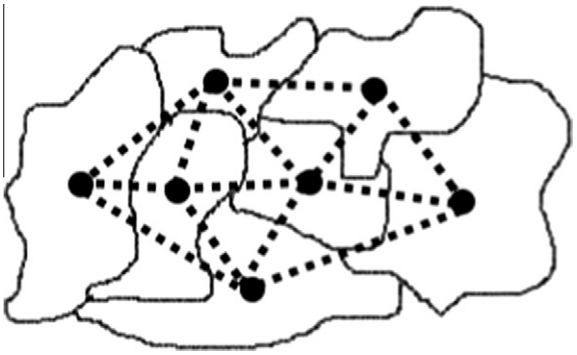

Fig. 2. The dual of the map.

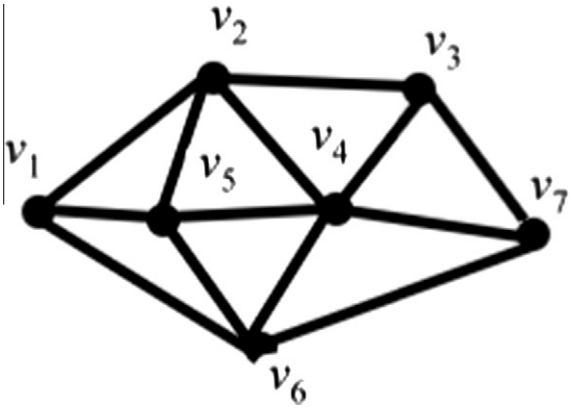

Fig. 3. The connected planar graph G.

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Fig. 4. The adjacency matrix A for Fig. 3.

Step 3: Specifying color number to the vertex

Coloring program of the adjacency matrix A of planar graph with n nodes $V = v_1, v_2, v_3, \ldots, v_n$ can be indicated as the coloring sequence $R = r_1, r_2, r_3, \ldots, r_n$, where $r_x \in R$ $(1 \leqslant x \leqslant n)$ and $r_x \in \{0, 1, 2, 3\}$. According to our coding assumption above, this coloring program corresponding coloring sequence of graph G with 7 nodes for Fig. 3 can be expressed as $R = 0, 1, 2, 3, 0, 1, 2$.

In order to determine whether the coloring sequence R satisfies the conditions of the coloring program, we define the fitness function $f(R)$ and the conflict matrix Co, according to adjacency matrix as stated in Eqs. (4) and (5).

$$f(R) = \sum_{x=1}^{n} \sum_{y=1}^{n} \text{conflict}_{xy}, \tag{4}$$

$$\text{conflict}_{xy} = \begin{cases} a_{xy} & \text{if } r_x = r_y \quad \text{and} \quad x \neq y \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

where r_x, $r_y \in R(1 \leqslant x \leqslant n, 1 \leqslant y \leqslant n)$,$a_{xy} \in A$, and $conflict_{xy}$ represents the coloring conflict between node v_x and node v_y, $f(R)$ represents the aggregate of node coloring in coloring sequence R. For example, the coloring sequence of graph G with 7 nodes as shown in Fig. 3 can be expressed as $R = 0, 1, 2, 3, 0, 1, 2$ as described above. In R, the $r_1 = 0$ and $r_5 = 0$. And in Fig. 4, the $a_{15} = 1$. Then the $conflict_{15} = 1$ by invoking Eq. (5). By the method, the $conflict_{12} = 0$, $conflict_{13} = 0$, ..., and then the conflict matrix Co for Fig. 3 is shown in Fig. 5 and the fitness function $f(R) = 4$.

Step 4: Adjust coloring number according to adjacency matrix

Obviously, the fitness function $f(R) \geqslant 0$ and $f(R)$ is an even number. In order to consider in such a way that no two adjacent vertices (i.e. regions) are of the same color, it corresponds to a reasonable program when the fitness function $f(R) = 0$. For $f(R) = 0$, the vertex must adjust the color number. For examples, we can change the coloring number r_1 of the vertex v_1 from 0 to 3 and the coloring number r_7 of the vertex v_7 from 2 to 0. Then the fitness function $f(R)$ will equal to 0 and the coloring sequence $R = 3, 1, 2, 3, 0, 1, 0$. Finally, the map coloring program will get solution as shown in Fig. 6.

3. Particle swarm optimization

The particle swarm optimization (PSO) is a novel multi-agent optimization system (MAOS) inspired by social behavior metaphor developed by Kennedy and Eberhart (1995, 1995, 1998). It is unlike an evolutionary algorithm, however, in that each potential solution is also assigned a randomized velocity, and the potential solutions, called particles, are then "flown" through the problem hyperspace. The PSO method was generally found to perform better than other algorithms (for example, genetic algorithm, memetic algorithm, antcolony systems, and shuffled frog leaping) in terms of success rate and solution quality (Eberhart et al., 1998; Elbeltagi, Hegazy, & Grierson, 2005). In PSO, instead of using more traditional genetic operators, each particle's velocity (individual) is stochastically accelerated toward its previous best position (where it had its best fitness value) and toward a neighborhood best position (the position of best fitness by any particle in its neighborhood).

3.1. Standard particle swarm optimization

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:

$$V_i = \omega \times V_i + c_1 \times rand_1() \times (P_{best} - P_i) + c_2 \times rand_2() \times (P_{Gbest} - P_i), \quad (6)$$

where c_1 and c_2 are positive acceleration constants (usually $c_1 = c_2 = 2$), $rand_1()$ and $rand_2()$ are uniformly distributed random numbers in the range [0,1], and ω is an inertia weight employed as an improvement proposed by Shi and Eberhart (1998). Then the particle flies toward a new position according to Eq. (7).

$$P_i = P_i + V_i. \quad (7)$$

The whole running procedure of the standard PSO is described in Algorithm 1.

$$Co = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig. 5. The conflict matrix Co for Fig. 3.

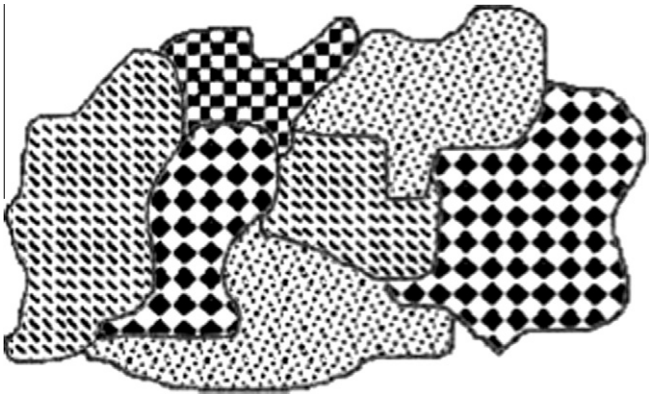

Fig. 6. The map coloring for Fig. 1.

Algorithm 1 (*Standard PSO algorithm*).

1. initialize all particles' positions P_i and velocities V_i, for $1 \leqslant i \leqslant$ 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, $(1 \leqslant i \leqslant$ NumberOfParticles) **do**
4. calculate the fitness value of particle i.
5. update the personal best position of particle i according to the fitness value.
6. **end for**
7. update the global best position of all particles according to the fitness value.
8. **for** particle i, $(1 \leqslant i \leqslant$ NumberOfParticles)**do**
9. move particle i to another position according to Eqs. (6) and (7).
10. **end for**
11. **end while**

3.2. Discrete particle swarm optimization

Since PSO was first introduced to optimize various continuous nonlinear functions by Kennedy and Eberhart (1995, 1995, 1998), it has been successfully applied to a wide range of applications. However, the major obstacle of successfully applying a PSO is its continuous real numbers, but it cannot be used to discrete problem directly. Aiming at this drawback, Kennedy and Eberhart (1997) developed a discrete binary version of PSO (named BPSO). In BPSO, the particle is characterized by a binary solution representation, and the velocity must be transformed into the change of probability for

each binary dimension to take a value of one. In 2008, Cui et al. (2008) 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. The model can improved BPSO and combined the solution requirement of the planar graph coloring problem to demonstrate the efficiency of PSO in discrete optimization problem. For assigning a new particle position, the quaternary-valued PSO method uses Eqs. (8) and (9) to replace Eq. (7).

$$f(V_j) = \begin{cases} 0, rand() > r \& rand() < S(V_j), \\ 1, rand() < r \& rand() < S(V_j), \\ 2, rand() \geqslant r \& rand() \geqslant S(V_j), \\ 3, rand() \leqslant r \& rand() \geqslant S(V_j), \end{cases} \tag{8}$$

$$P_j = Mod((P_j + f(V_j)), 4), \tag{9}$$

where $V_j (1 \leqslant j \leqslant \text{number of nodes})$ is particle's velocity, $rand()$ is a uniformly distributed random number in the range $[0,1]$, $r = 0.5$, $S(V_j)$ is the sigmoid function given by $S(v) = 1/(1 + e^{-v})$ and Mod(number, divisor) function returns the remainder after a number is divided by a divisor.

4. The new proposed planar graph coloring model

A new planar graph coloring, named MTPSO, based on the particle swarm optimization. The MTPSO model, which employs the walking one strategy, assessment strategy and turbulent strategy, is proposed in this paper.

4.1. Walking one strategy

Cui et al. (2008) 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. (10) based on Eq. (5). Then the number of the conflict nodes will be converted to collision factor through the sigmoid function (i.e. Eq. (11)). 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. (12) and (13)). The walking one strategy is shown as follows:

$$Cr_j = \sum_{k=1}^{n} \text{conflict}_{jk}, \tag{10}$$

$$Cf_j = \frac{1}{1 + e^{-Cr_j + 2}}, \tag{11}$$

$$M(V_j) = \begin{cases} 1, & \text{if } Cf_j > rand() \& S(V_j) > rand(), \\ 0, & \text{otherwise}, \end{cases} \tag{12}$$

$$P_j = Mod((P_j + M(V_j)), 4), \tag{13}$$

where $Cr_j (1 \leqslant j \leqslant n)$ is the number of conflict nodes with jth node, and Cf_j is the collision factor of jth node in the range $[0,1]$, n is the number of nodes, $rand()$ is a uniformly distributed random number in the range $[0,1]$, $S(V_j)$ is the sigmoid function given by $S(v) = 1/(1 + e^{-v})$. For example, in Fig. 5, the $Cr_1 = 1$ and $Cr_2 = 0$ by invoking Eq. (10), then the $Cf_1 = 0.27$ and $Cf_2 = 0.12$ by invoking Eq. (11). Suppose the particle velocity $V_1 = 1$ and $V_2 = 1$, then the $S(V_1) = 0.73$ and $S(V_2) = 0.73$. And suppose $rand_1 = 0.2$ and $rand_2 = 0.2$, then the $M(V_1) = 1$ and $M(V_2) = 0$ by invoking Eq. (12). If the current position of particle $P_1 = 1$ and $P_2 = 1$, then the next position of particle P_1 will move to 2, but P_2 will stay in 1.

4.2. Assessment strategy

In graph coloring, the maximum conflict node is the most troublesome and needs to be processed first. In this strategy, we will find the maximum conflict node and assess it. The strategy is described in Algorithm 2:

Algorithm 2 (*Assessment strategy algorithm*).

1. **set** *fgChanged* = **False**
2. **for** color o, $(0 \leqslant o \leqslant 3)$ **do**
3. $P_{\max(Cr_j)} = o (1 \leqslant j \leqslant n)$
4. calculate the new fitness value of particle i (i.e., f_{ni}).
5. **if** $f_{ni} < f_{oi}$ **then**
6. **set** *fgChanged* = **True**
7. exit for;
8. **end if**
9. **end for**

where $P_{\max(Cr_j)}$ is the maximum conflict node of particle, and $Cr_j (1 \leqslant j \leqslant n)$ is the number of conflict nodes with jth node (i.e. Eq. (10)). f_{ni} represents new fitness value with changing the maximum conflict node of particle i. f_{oi} represents old fitness value of particle i. The flag *fgChanged* represents the changing status of maximum conflict node of particle.

4.3. Turbulent strategy

In the standard PSO, whether it is in continuous or discrete, it has been shown that the trajectories of the particles oscillate in different sinusoidal waves and converge quickly, sometimes prematurely (Liu & Abraham, 2005). Such situations could occur even in the early stages of the search. In fact, this does not even guarantee that the algorithm has converged to a local minimum and it merely means that all the particles have converged to the best position discovered so far by the swarm. During each iteration, the particle is attracted toward the location of the best fitness achieved so far by the particle itself and by the location of the best fitness achieved so far across the whole swarm. In order to guide the particles effectively in the search space, the maximum moving distance during an iteration and its moving range must be clamped in between the maximum velocity. The method to drive those lazy particles (i.e. the particles' velocity is smaller than threshold) and so that they can explore a better solution, named turbulent particle swarm optimization (TPSO) which is shown as follows:

$$V_j = \begin{cases} -1 + rand() \times 2 & \text{if } |V_j| < V_s, \\ V_j & \text{otherwise}, \end{cases} \tag{14}$$

where $V_j (1 \leqslant j \leqslant n)$ is the velocity of jth node, n is the number of nodes, $rand()$ is a uniformly distributed random number in the range $[0,1]$, V_s is the minimum velocity threshold, a threshold parameter to limit the minimum of the particles' velocity. If V_s is large, it will shorten the oscillation period, and it provides a great probability for the particles to across local optimal using the same number of iterations. But a large V_s compels particles in the quick "flying" state, which leads them not to search the local optimal and forcing them not to refine the search. In other words, a large V_s facilitates a global search, and a small V_s facilitates a local search instead.

4.4. MTPSO algorithm

The whole running procedure of the MTPSO is described in Algorithm 3.

Algorithm 3 (*MTPSO algorithm*).

1. initialize all particles' positions (i.e. coloring number) P_i and velocities V_i, for $1 \leqslant i \leqslant$ NumberOfParticles.
2. **while** the stop condition (the optimal solution is found or the maximal moving steps are reached) is not satisfied **do**
3. change the inertia weight ω according to the number of iterations.
4. **for** particle i, $(1 \leqslant i \leqslant$ NumberOfParticles) **do**
5. calculate f_{oi} the old fitness value of particle i.
6. update the personal best position of particle i according to the fitness value.
7. **end for**
8. update the global best position of all particles according to the fitness value.
9. **for** particle i, $(1 \leqslant i \leqslant$ NumberOfParticles) **do**
10. calculate the velocity of particle i according to Eq. (6).
11. turbulent the velocity of particle i according to Eq. (14).
12. assessment whether to accept the change of the maximum conflict node with the fitness value according to Algorithm 2.
13. **if** *fgchanged* = **True then**
14. move particle i to another position with the change of the maximum conflict node.
15. **else**
16. move particle i to another position according to Eqs. (10)–(13).
17. end if
18. **end for**
19. **end while**

An example for illustrating the MTPSO model for solving planar graph coloring problem is given in the following. Initially, the value of the velocity threshold V_s corresponding to the former is set to 0.00001, both c_1 and c_2 are set to 2, and ω is set to 2. Let the number of particles be 5. A particle is defined as a vector consisting of n elements (i.e. $b_1, b_2, \ldots, b_{j-1}, b_j, \ldots, b_{n-1}, b_n$, where $1 < j \leqslant n$). That is, we base on these n elements to define the n nodes (i.e. regions in the map). For simplicity, please refer to Section 2 for the continuation of the example of Fig. 1 in detail. The randomized initial positions and the initial velocities of all particles are listed in Tables 1 and 2, respectively. And the numbers of conflict nodes with each node of all particles are listed in Table 3.

After all particles have got their own fitness values, every particle updates its own personal best position so far according to the fitness value. Note that the initial personal best positions are set as the initial positions of all particles. The personal best positions of all particles so far are listed in Table 1. In Table 1, since the fitness value of particle 5 is the least among all five particles so far, the global best position is then set to particle 5. Then the MTPSO algorithm moves all particles to the second positions according to Eqs. (6) and (10)–(14).

$$
\begin{aligned}
V_{b1} &= 2 \times (-2.812) + 2 \times rand_1() \times (0-0) \\
&\quad + 2 \times rand_2() \times (1-0) = -5.022, \\
V_{b2} &= 2 \times (2.243) + 2 \times rand_1() \times (3-3) \\
&\quad + 2 \times rand_2() \times (2-3) = 3.885, \\
V_{b3} &= 2 \times (1.025) + 2 \times rand_1() \times (1-1) \\
&\quad + 2 \times rand_2() \times (0-1) = 1.449, \\
V_{b4} &= 2 \times (-3.738) + 2 \times rand_1() \times (2-2) \\
&\quad + 2 \times rand_2() \times (2-2) = -7.475, \\
V_{b5} &= 2 \times (-3.812) + 2 \times rand_1() \times (3-3) \\
&\quad + 2 \times rand_2() \times (1-3) = -8.828, \\
V_{b6} &= 2 \times (-2.860) + 2 \times rand_1() \times (2-2) \\
&\quad + 2 \times rand_2() \times (3-2) = -5.118, \\
V_{b7} &= 2 \times (-2.622) + 2 \times rand_1() \times (1-1) \\
&\quad + 2 \times rand_2() \times (1-1) = -5.244.
\end{aligned}
\tag{15}
$$

It makes no difference for the Cr_{b2} even if the maximum conflict node P_{b2} uses any one of the colors. Hence, in the assess stage of the particle 1, the numbers of conflict nodes vector $Cr = \{0,1,1,1,1,1,1\}$ is not changed. Then the particle 1 enters walking one strategy. In walking one strategy, the velocity of particle 1 is calculated in Eq. (15) based on Eq. (6), and the velocity vector is $V = \{-5.022, 3.885, 1.449, -7.475, -8.828, -5.118, -5.244\}$. And then it will be translated to $\{0.0065, 0.9799, 0.8098, 0.0006, 0.0001, 0.0060, 0.0053\}$ by using sigmoid function. The absolute value velocity vector is larger than V_s(i.e. Eq. (14)), so the turbulent strategy will not operate. And the Cr vector is $\{0,1,1,1,1,1,1\}$ by using Eqs. (5) and (10), and the Cf vector equals to $\{0.1192, 0.2689, 0.2689, 0.2689, 0.2689, 0.2689, 0.2689\}$ by using Eq. (11). The *rand* vector is $\{0.51, 0.23, 0.89, 0.45, 0.61, 0.28, 0.15\}$. And the $M(V) = \{0,1,0,0,0,0,0\}$ by using Eq. (12). Then the new position vector of particle 1 is $\{0,0,1,2,3,2,1\}$ based on Eq. (13).

The particle 2 in assessment strategy (i.e. Algorithm 2), the maximum conflict node P_{b6} will change color from 1 to 0, and Cr_{b6} will down to 0. And the numbers of conflict nodes vector Cr will change from $\{1,0,0,2,0,3,2\}$ to $\{0,0,0,2,0,0,2\}$. And changed flag *fgChanged* will set to **False**. Then the new position vector of particle 2 equals to $\{1,2,3,1,0,0,1\}$.

Table 1
The initial position of particles 1–5.

	b_1	b_2	b_3	b_4	b_5	b_6	b_7	Fitness value
particle1	0	3	1	2	3	2	1	6
particle2	1	2	3	1	0	1	1	8
particle3	3	1	2	1	3	1	2	8
particle4	2	2	3	3	0	0	1	6
particle5	1	2	0	2	1	3	1	4

Table 2
The initial velocities of particles 1–5.

	b_1	b_2	b_3	b_4	b_5	b_6	b_7
particle1	−2.812	2.243	1.025	−3.738	−3.812	−2.860	−2.622
particle2	2.614	−3.089	3.124	−0.503	−1.350	0.357	−1.151
particle3	−0.891	2.426	−3.674	−0.496	−1.076	1.922	−2.531
particle4	−3.117	2.967	−0.434	−2.170	2.675	3.084	−2.687
particle5	2.849	−2.736	2.019	−3.562	2.066	−2.254	−0.095

Table 3
The numbers of conflict nodes with each node of particles 1–5.

	b_1	b_2	b_3	b_4	b_5	b_6	b_7	Fitness value
particle1	0	1	1	1	1	1	1	6
particle2	1	0	0	2	0	3	2	8
particle3	1	1	1	2	1	1	1	8
particle4	1	1	1	1	1	1	0	6
particle5	1	1	0	1	1	0	0	4

Table 4
The second positions of particles 1–5.

	b_1	b_2	b_3	b_4	b_5	b_6	b_7	Fitness value
particle1	0	0	1	2	3	2	1	6
particle2	1	2	3	1	0	0	1	4
particle3	3	1	2	0	3	1	2	4
particle4	1	2	3	3	0	0	1	4
particle5	0	2	0	2	1	3	1	2

Table 5
The second velocities of particles 1–5.

	b_1	b_2	b_3	b_4	b_5	b_6	b_7
particle1	−5.022	3.885	1.449	−7.475	−8.828	−5.118	−5.244
particle2	5.228	−6.177	1.279	0.651	−1.044	4.028	−2.301
particle3	−4.886	6.404	−10.453	0.560	−5.257	6.950	−6.615
particle4	−7.799	5.934	−5.561	−5.903	6.914	10.859	−5.373
particle5	5.698	−5.473	4.037	−7.124	4.132	−4.508	−0.189

Table 6
Comparison of the experimental results[a].

Node n	Algorithms	Maximal iterations	Minimal iterations	Average iterations	Correct coloring rate
10	MTPSO	1	0	0.16	100%
	MPSO	3	0	0.35	100%
15	MTPSO	2	0	1.47	100%
	MPSO	32	0	6.5	100%
20	MTPSO	3	0	1.74	100%
	MPSO	575	1	135.04	100%
25	MTPSO	6	2	4.36	100%
	MPSO	7202	63	1956.79	100%
30	MTPSO	7	2	4.68	100%
	MPSO	10000	1789	9497.78	11%
50	MTPSO	15	7	11.48	100%
	MPSO	10000	10000	10000	0%
100	MTPSO	42	23	29.83	100%
	MPSO	10000	10000	10000	0%
150	MTPSO	125	46	73.92	100%
	MPSO	10000	10000	10000	0%
200	MTPSO	249	53	127.68	100%
	MPSO	10000	10000	10000	0%
250	MTPSO	582	88	298.88	100%
	MPSO	10000	10000	10000	0%

[a] MTPSO is the algorithm proposed in this paper, and MPSO is the algorithm proposed in Cui et al. (2008).

By these procedures, we can get the position and velocity of each of the other particles. Now the MTPSO model moves all particles to the second positions according to Algorithm 2, Eqs. (6) and (10)–(14). The second positions and the corresponding new fitness values of all particles are listed in Table 4.

By comparing the fitness values listed in Table 1 with those listed in Table 4, it is obvious that particles 2–5 reach a better position than its own personal best position so far. Also the new global best position for all particles is created by particle 5 as its fitness is the least among all particles so far. The MTPSO model repeats the above steps until the stop condition is satisfied.

5. Experimental results

The essential parameters of MTPSO model for the planar graph coloring are set as follows. We simulated 100 runs in each order. Let the maximal number of iterations be 10000, the number of particles be 200, the value of inertial weight (i.e. ω) be 2 linear decreased by iterations to 0.3, the self confidence coefficient (i.e. c_1) and the social confidence coefficient (i.e. c_2) both be 2, the minimum velocity threshold V_s be 0.00001, respectively.

In order to illustrate the performance of the algorithms presented in this paper, we implemented the algorithm presented in Cui et al. (2008). The essential parameters of modified PSO for the planar graph coloring are set as follows. We simulated 100 runs in each order. Let the maximal number of iterations be 10000, the number of particles be 200, the value of inertial weight (i.e. ω) be 2 linear decreased by iterations to 0.8, the self confidence coefficient (i.e. c_1) be 2 and the social confidence coefficient (i.e. c_2) be 1.8, the disturbance factor be 5, respectively.

By randomly generating a given scaled planar graph, and separately calculating 100 times to MTPSO algorithm and modified PSO (i.e. MPSO), a comparison of the planar graph coloring efficiency (i.e. the average iterations) and accuracy (i.e. the correct coloring rate) between the proposed model and Cui et al. (2008) model is listed in Tables 5 and 6. We can see that the proposed model gets smaller average iterations than the model presented in Cui et al. (2008). Also the proposed model gets the higher correct coloring rate than the model presented in Cui et al. (2008) when the number of nodes is grater than 30.

6. Conclusions

In this paper, we proposed a modified turbulent particle swarm optimization (named MTPSO) for solving planar graph coloring based on particle swarm optimization. The proposed model is consisting of the walking one strategy, assessment strategy and turbulent strategy for improving the discrete PSO in the planar graph coloring. The walking one strategy and assessment strategy can improve the performance of MTPSO algorithm. The turbulent strategy can be helpful to drive those lazy particles (i.e., those velocities of the particles smaller than threshold) and hence they can explore a better solution. The experimental results show that the MTPSO model is more efficient and accurate than the modified PSO algorithm proposed by Cui et al. (2008).

References

Blochliger, I., & Zufferey, N. (2008). A graph coloring heuristic using partial solutions and a reactive tabu scheme. *Computers & Operations Research, 35*, 960–975.

Bui, T. N., Nguyen, T. H., Patel, C. M., & Phan, K.-A. T. (2008). An ant-based algorithm for coloring graphs. *Discrete Applied Mathematics, 156*, 190–200.

Chaitin, G. J., Auslander, M. A., Chandra, A. K., Cocke, J., Hopkins, M. E., & Markstein, P. W. (1981). Register allocation via coloring. *Computer Languages, 6*, 47–57.

Chow, F. C., & Hennessy, J. L. (1990). Priority-based coloring approach to register allocation. *ACM Transactions on Programming Languages and Systems, 12*, 501–536.

Cui, G., Qin, L., Liu, S., Wang, Y., Zhang, X., & Cao, X. (2008). Modified PSO algorithm for solving planar graph coloring problem. *Progress in Natural Science, 18*, 353–357.

de Werra, D. (1985). An introduction to timetabling. *European Journal of Operational Research, 19*, 151–162.

Dowsland, K. A., & Thompson, J. M. (2008). An improved ant colony optimisation heuristic for graph colouring. *Discrete Applied Mathematics, 156*, 313–324.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Proceedings of the sixth international symposium on micro machine and human science MHS '95* (pp. 39–43).

Eberhart, R. C., & Shi, Y. (1998). *Comparison between genetic algorithms and particle swarm optimization* (Vol. 1447/1998). Berlin/ Heidelberg: Springer.

Elbeltagi, E., Hegazy, T., & Grierson, D. (2005). Comparison among five evolutionary-based optimization algorithms. *Advanced Engineering Informatics, 19*, 43–53.

Galinier, P., & Hertz, A. (2006). A survey of local search methods for graph coloring. *Computers & Operations Research, 33*, 2547–2562.

Gamst, A. (1986). Some lower bounds for a class of frequency assignment problems. *IEEE Transactions on Vehicular Technology, 35*, 8–14.

Garey, M. R., Johnson, D. S., & So, H. C. (1976). Application of graph coloring to printed circuit testing. *IEEE Transactions on Circuits Systems, CAS-23*, 591–599.

Kennedy, J. & Eberhart, R. 1995. Particle swarm optimization. In Proceedings of IEEE international conference on neural networks (Vol. 4, pp. 1942–1948).

Kennedy, J., & Eberhart, R.C. (1997). A discrete binary version of the particle swarm algorithm. In IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation (Vol. 5, pp. 4104–4108).

Kuo, I. H., Horng, S.-J., Kao, T.-W., Lin, T.-L., Lee, C.-L., & Pan, Y. (2009). An improved method for forecasting enrollments based on fuzzy time series and particle swarm optimization. *Expert Systems with Applications, 36*, 6108–6117.

Kuo, I. H., Horng, S.-J., Kao, T.-W., Lin, T.-L., Lee, C.-L., Terano, T., et al. (2009). An efficient flow-shop scheduling algorithm based on a hybrid particle swarm optimization model. *Expert Systems with Applications, 36*, 7027–7032.

Leighton, F. T. (1979). Graph coloring algorithm for large scheduling problems. *Journal of Research of the National Bureau of Standards (United States), 84*, 489–506.

Liu, H. & Abraham, A. (2005). Fuzzy adaptive turbulent particle swarm optimization. In *Fifth international conference on hybrid intelligent systems, HIS '05*, 6 pp.

Osman, I. H. (2006). A tabu search procedure based on a random Roulette diversification for the weighted maximal planar graph problem. *Computers & Operations Research, 33*, 2526–2546.

Park, J.-I., Lee, D.-J., Song, C.-K., & Chun, M.-G. (2010). TAIFEX and KOSPI 200 forecasting based on two-factors high-order fuzzy time series and particle swarm optimization. *Expert Systems with Applications, 37*, 959–967.

Salari, E. & Eshghi, K. (2005). An ACO algorithm for graph coloring problem. *In 2005 ICSC Congress on computational intelligence methods and applications*, 5 pp.

SangHyuck, A., SeungGwan, L., & TaeChoong, C. (2003). Modified ant colony system for coloring graphs. In *Proceedings of the 2003 joint conference of the fourth international conference onInformation, communications and signal processing, 2003 and the fourth pacific rim conference on multimedia* (Vol. 3, pp. 1849–1853.

Shi, Y. & Eberhart, R.C. (1998). A modified particle swarm optimizer. *In IEEE International conference on evolutionary computation, proceedings of IEEE world congress on computational intelligence* (pp. 69–73).

Talavan, P. M., & Yanez, J. (2008). The graph coloring problem: A neuronal network approach. *European Journal of Operational Research, 191*, 100–111.

Yanez, J., & Ramirez, J. (2003). The robust coloring problem. *European Journal of Operational Research, 148*, 546–558.