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 Probrem; 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.

$$v_i^{k+1} = wv_i^k + c_1 r_1 (pbest_i^k - x_i) + c_2 r_2 (gbest^k - x_i). \quad (1)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1}. \quad (2)$$

978-1-4799-8697-2/15 $31.00 © 2015 IEEE
DOI 10.1109/SMC.2015.341

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 $n/3$ 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 $d = m/n$. 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.

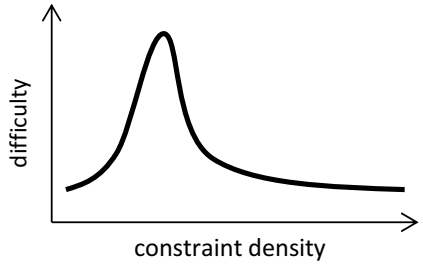

Figure 1. Image of the relationship between constraint density and difficulty

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 $x_i = (x_{i1}, x_{i2}, x_{i3}, \ldots x_{ij}, \ldots x_{in})$, 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).

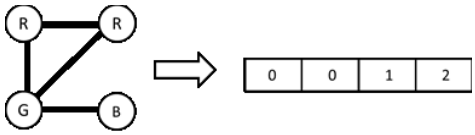

Figure 2. An example of GCP

C. Related Works

1) MTPSO: Hsu et al. proposed a modified turbulent particle swarm optimization (MTPSO) model baced 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_{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} ($P_{own}+P_{pbest}+P_{gbest}=1$).

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].

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

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

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

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 *pbest* or *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 $pbest_i^k$ or $gbest^k$ is, the bigger the transition probability changing to $pbest_i^k$ or $gbest^k$ 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} ($P_{rand} + P_{pbest} + P_{gbest}=1$); 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.

$$x \ominus y = 1 - \frac{H(x,y)}{n} \quad (6)$$

$$v_i^k = (x_i^k \ominus x_i^{k-1}) \quad (7)$$

$$V_{rand} = w v_i^k \quad (8)$$

$$V_{pbest} = c_1 r_1 (x_i^k \ominus pbest_i^k) \quad (9)$$

$$V_{gbest} = c_2 r_2 (x_i^k \ominus gbest^k) \quad (10)$$

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

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

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

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

$$x_{ij}^k = \begin{cases} random\ color & (P_{rand}) \\ pbest_{ij}^k & (P_{pbest}) \\ gbest_j^k & (P_{gbest}) \end{cases} \quad (15)$$

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_i;
      }
      Update gbest;
}

Procedure update-position(x_i){
      for ( j = 1 to N) {
        Generate number r ∈ [0, 1];
        if(r <= P_rand){x_ij = random color;}
        else if(rand <= P_rand+ P_pbest){
            x_ij = pbest_ij;
        }
        else{
            x_ij = gbest_j;
        }
      }
}
```

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_x 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 1. Experimental conditions

	Exp. 1	Exp. 2	Exp. 3
Population size	10	10	10
Max iteration	1.0×10^5	1.0×10^5	1.0×10^5
w	-	E_w	E_w
c_1	1.0	1.0	-
c_2	1.0	-	E_{c2}
Result	Fig. 3	Fig. 4	Fig. 5

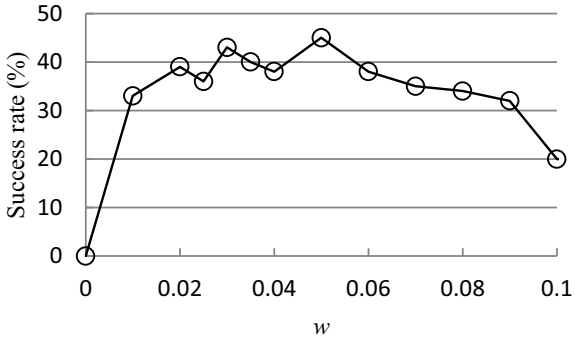

Figure 3. Relationship between success rate and w

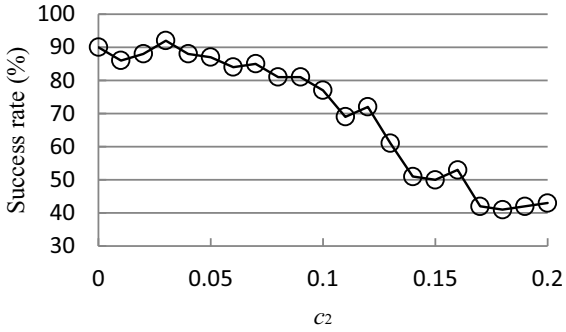

Figure 4. Relationship between success rate and c_2

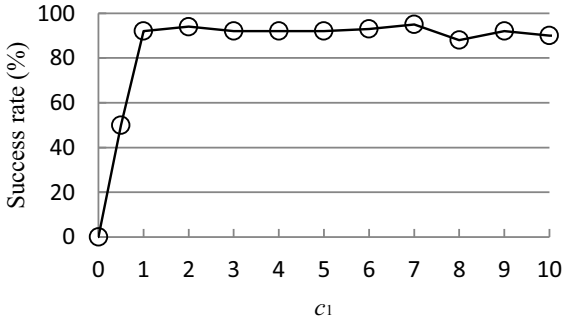

Figure 5. Relationship between success rate and c_1

Second, we examined searcing 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.

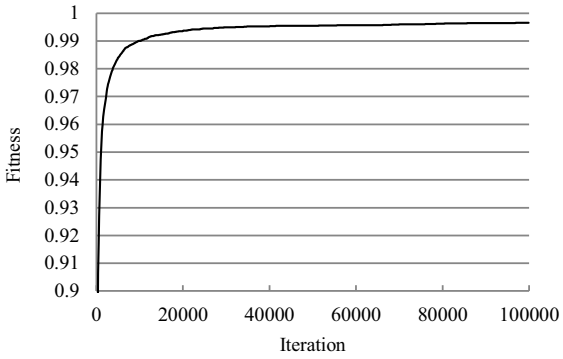

Figure 6. Searching progress

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 2. Experimental conditions

	Proposed	TPPSO	GA
Population size	10	10	50
Max iteration	1.0×10^5	1.0×10^5	2.0×10^4
w	0.05	0.02	-
c_1	7.0	2.5	-
c_2	0.03	1.0	-
Mutation rate	-	-	1.4%
Tournament size	-	-	2
Elite size	-	-	1

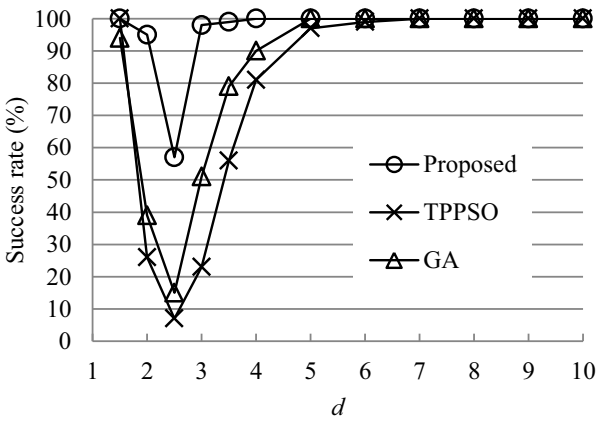

Figure 7. Comparative study of success rate (n=90)

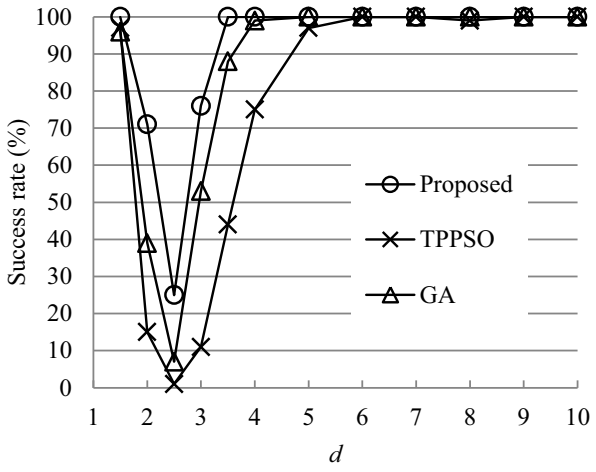

Figure 8. Comparative study of success rate (n=120)

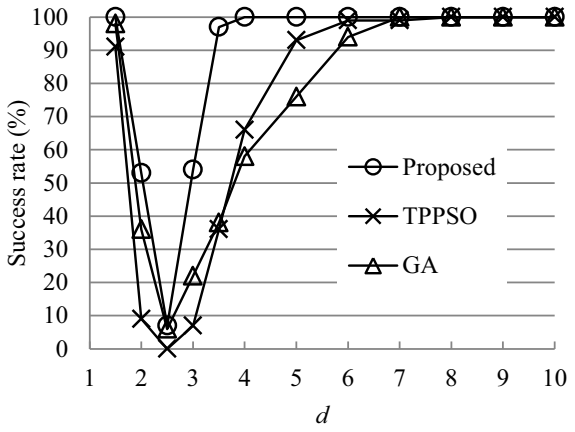

Figure 9. Comparative study of success rate (n=150)

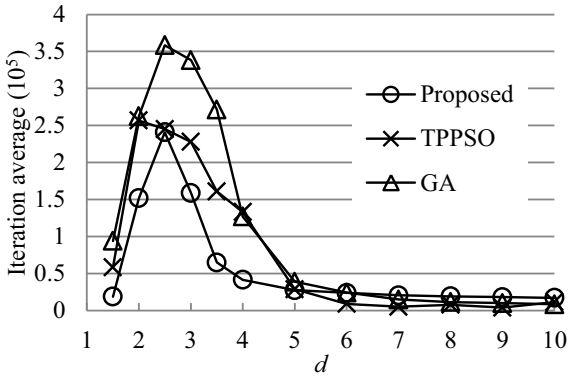

Figure 10. Comparative study of Iteration average (n=90)

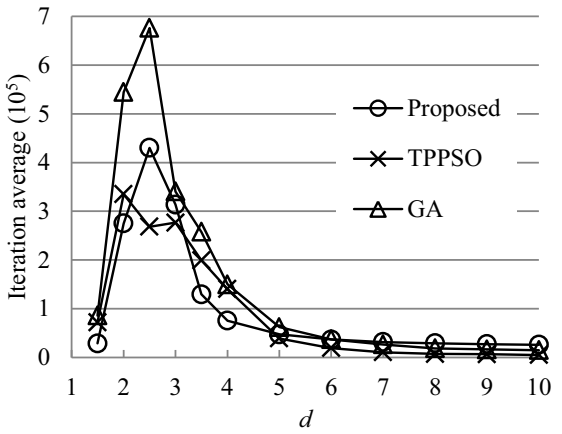

Figure 11. Comparative study of Iteration average (n=120)

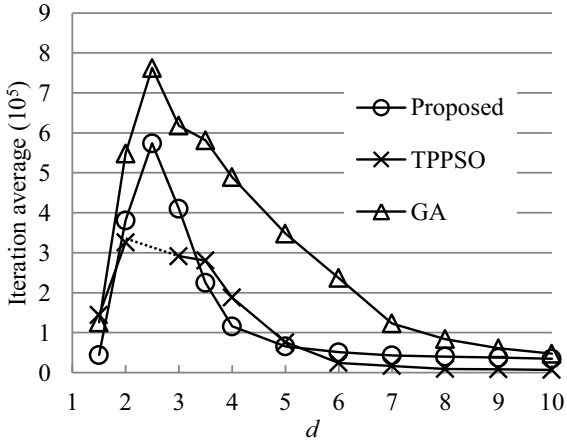

Figure 12. Comparative study of Iteration average (n=150); because no solution can be found at d=2.5, the line is indicated by dotted line.

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 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 *pbest* and *gbest* are more effective in searching when $d < 6.5$ and d > 6.5, respectively.

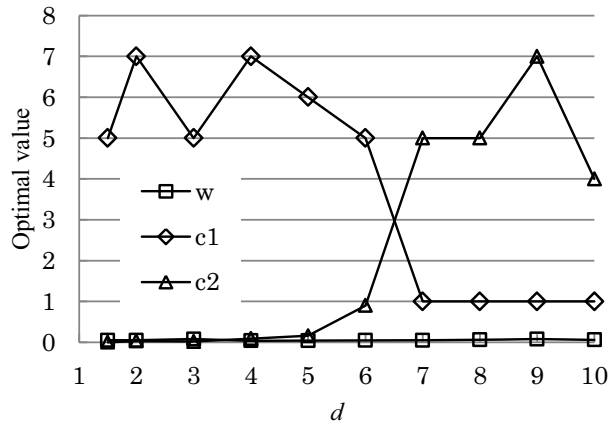

Figure 13. Dependence of the parameters on constraint density

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

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

2) Jonas Krause, Jelson Cordeiro1, Rafael Stubs Parpinelli, Heitor Silve´rio Lopes.: A Survey of Swarm Algorithms Applied to Discrete Optimization Problems, Swarm Intelligence and Bio-Inspired Computation, pp.169-191, 2013.

3) Kennedy, J., Eberhart R.C.: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, 1997.

4) Deep, K., Bansal, J.C. : A socio-cognitive particle swarm optimization for multidimensional knapsack problem. In: First International Conference on Emerging Trends in Engineering and Technology, pp. 355-360, 2008.

5) Hembecker, F., Lopes, H.S., Godoy Jr., W. : . Particle swarm optimization for the multidimensional knapsack problem. Proceedings of the Eighth International Conference on Adaptive and Natural Computing Algorithms, Part I. Vol. 4431 of Lecture Notes in Computer Science. Springer-Verlag, Heidelberg, pp. 358-365, 2007.

6) Shen, X., Li, Y., Chen, C., Yang, J., Zhang, D. : Greedy continuous particle swarm optimisation algorithm for the knapsack problems. Int. J. Comput. Appl. Technol. 44 (2), pp.137-144, 2012.

7) Ucar, H., Tasgetiren, M.F.: A particle swarm optimization algorithm for permutation flow shop sequencing problem with the number of tardy jobs criterion. In: Fifth International Symposium on Intelligent Manufacturing Systems, pp. 237-250, 2006.

8) Rosendo, M., Pozo, A.: Applying a discrete particle swarm optimization algorithm to combinatorial problems. Brazilian Symposium on Neural Networks. IEEE Computer Society, Los Alamitos, CA, pp. 235-240, 2010.

9) Guangzhao Cui, Limin Qin, Sha Liu, Yanfeng Wang, Xuncai Zhang, Xianghong Cao: Modified PSO algorithm for solving planar graph coloring problem, Progress in Natural Science, Vol.18, pp.353–357, 2008.

10) Ling-Yuan Hsu, Shi-JinnHorng, Pingzhi Fan, Muhammad Khurram, Yuh-Rau Wang, Ray-Shine Run, Jui-Lin Lai, Rong-Jian Chan: MTPSO algorithm for solving planar graph coloring problem, Expert Systems with Application,Vol.38, pp.5525-5531, 2011.

11) Hitoshi Kanoh, Satoshi Chen : Particle Swarm Optimization with Transition Probability for Timetabling Problems, Adaptive and Natural Computing Algorithms, Vol. 7824, pp 256-265 , 2013.

12) Wei-Neng Chen, Jun Zhang, Henry S. H. Chung, Wen-Liang Zhong, Wei-Gang Wu, Yu-hui Shi : A Novel Set-Based Particle Swarm Optimization Method for Discrete Optimization Problems, IEEE Transactions on evolutionary computation, Vol. 14, No. 2, April, pp.278-300, 2010.

13) Steven Minton, Mark D. Johnston, Andrew B. Philips, Philip Laird: Minimizing Conflicts: A Heuristic Repair Method for Constraint-Satisfaction and Scheduling Problems, Artificial Intelligence, Vol.58, pp.161-205, 1992.

14) Tad Hogg, Colin Williams: The Hardest Constraint Problems: A Double Phase Transition, Artificial Intelligence, Vol.69, pp.359-377, 1994.