Hybrid Discrete Particle Swarm Algorithm for Graph Coloring Problem

Jin Qin
College of Information Engineering, University of Science & Technology Beijing, Beijing, China
College of Computer Science & Information, Guizhou University, Guiyang, China
cse.jqin@gzu.edu.cn

Yi-xin Yin and Xiao-juan Ban
College of Information Engineering, University of Science & Technology Beijing, Beijing, China

***Abstract*—Graph coloring problem (GCP) is one of the most studied combinatorial optimization problems. It is important in theory and practice. There have been many algorithms for graph coloring problem, including exact algorithms and (meta-)heuristic algorithms. In this paper, we attempt another meta-heuristic method—particle swarm optimization to graph coloring problem. Particle swarm optimization (PSO) was originally developed for continuous problem. To apply PSO to discrete problem, the standard arithmetic operators of PSO are required to be redefined over discrete space. A conception of distance over discrete solution space is introduced. Under this notion of distance, the PSO operators are redefined. After reinterpreting the composition of velocity of a particle, a general discrete PSO algorithm is proposed. In order to solve graph coloring problem by the discrete PSO algorithm, an algorithm to implement the crucial PSO operator—difference of two positions (solutions) is designed. Then, a hybrid discrete PSO algorithm for graph coloring problem is proposed by combining a local search. Empirical study of the proposed hybrid algorithm is also carried out based on the second DIMACS challenge benchmarks. The experimental results are competitive with that of other well-known algorithms.**

***Index Terms*—graph coloring problem, DIMACS benchmarks, discrete particle swarm optimization, distance, redefinition of operators**

I. Introduction

The graph coloring problem (GCP) is one of the most studied combinatorial optimization problems. The challenge is, given a graph, to find the least number of colors for which there is a coloring of the vertices of the graph in which no two adjacent vertices bear the same color. The least number of colors needed to color a graph G is called its chromatic number, denoted $\chi(G)$. A graph that can be assigned a legal (conflict-free) k-coloring is k-colorable.

The problem of coloring a graph has found numerous applications including for instance, scheduling, register allocation in compilers and frequency assignment in mobile networks. Unfortunately, the problem is very difficult to solve. The problem of finding the chromatic number is NP-hard. The corresponding decision problem (Is there a coloring which uses at most k colors?) is NP-complete [1].

The problem is most often solved by using a conflict minimization algorithm. Given k colors, a coloring is sought which minimizes the number of conflicts (i.e., the number of adjacent vertices bearing the same color). Here we can find the sequential algorithms (those that color one vertex at a time, such as well-known DSATUR [2] and RLF [3]) and (meta-)heuristic algorithms (such as tabu search [4][5], simulated annealing [6], genetic algorithms [7], ant colony optimization [8], variable neighborhood search [9] and variable space search [10], etc.). In this paper, we attempt an adaptation of another meta-heuristic method—particle swarm optimization to graph coloring problem.

Particle swarm optimization (PSO) is a population-based search algorithm, originated from the simulation of the social behavior of birds within a flock. Since particle swarm optimization as a method for optimization of continuous nonlinear problem was introduced by James Kennedy and Russell Eberhart in 1995 [11], it has been used across a wide range of applications, such as image and video analysis, design and restructuring of electricity networks, control, antenna design, electronics and electromagnetics, and so on [12].

Particle swarm optimization was originally developed for continuous-valued spaces. The simplest and necessary change to the PSO for discrete problems is discretization of the position vectors. For more complex discrete problems over combinatorial spaces, the standard arithmetic operators used in the velocity and position equations, including addition, subtraction and multiplication, need to be redefined. Generally, these changes also change the interpretation of velocity, particle trajectory, velocity clamping and momentum [13].

There have been a few attempts to extend PSO to discrete spaces. Kennedy and Eberhart developed a discrete binary version of PSO that operates on binary space [14]. Clerc designed a discrete PSO for solving traveling salesman problem [15]. Moraglio et al. generalized PSO to any type of solution space by introducing a topological/ geometric crossover [16]. In [17], the original PSO operator was generalized in a

doi:10.4304/jcp.6.6.1175-1182

formal manner to permutation problem domain using forma analysis in form of equivalence relations.

In this paper, inspired by the idea in [16], we redefine the standard operators of PSO and reinterpret the components of velocity of a particle for optimization of discrete problems. And then we propose a hybrid discrete PSO algorithm for graph coloring problem by combining a local search. Empirical study of the proposed hybrid algorithm is also carried out based on the DIMACS benchmarks.

II. Generalization of PSO to Discrete Space

A. Notion of Distance over Discrete Solution Space

The generalization of PSO to discrete space involves redefining the standard PSO operators over discrete space. Taking into account the basic idea of PSO, the redefinition of the operators must manage to embody a true distance in the search space. A true distance should be a good measure of difference of two points in the search space. Based on the work in [16], we introduce a notion of distance which is a good foundation for discrete PSO.

Definition 1 (Solution space) A (discrete) solution space is a pair (S,o) where S is a set of solutions and o is an operator which operates on an element in S and generates another one. The operator must be reversible and connected (any solution can be transformed into any other by applying the operator a finite number of times).

A (discrete) solution space (S,o) is associated with (or represented by) an undirected graph (known as state-space graph in most artificial intelligence literatures) $G=(V,E)$ where $V=S$ and $E=\{(s_i,s_j)\mid s_i,s_j\in S, o(s_i)=s_j\}$. A vertex in the graph G represents a solution in S and an edge represents an application of operator which operates on one of endpoints of the edge to produce the other endpoint of the edge. For example, a set of binary strings with length 3 and an operator which reverses one bit of a binary string from 0 to 1 or 1 to 0 constitutes a solution space. Fig. 1 illustrates its associated state-space graph.

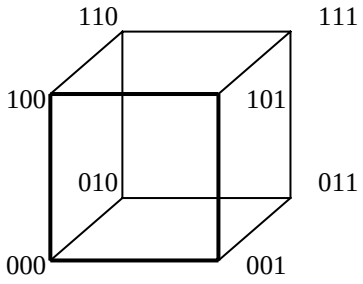

Fig. 1 An example of state-space graph with which a solution space is associated

Definition 2 (Metric space) A metric space is a pair (M,d) where M is a set and d is a metric or distance on M, i.e., a function $d: M\times M\to R$ such that for any x, y and z in M

(1) $d(x,y)=0$ if and only if $x=y$; (identity)

(2) $d(x,y)=d(y,x)$; (symmetry)

(3) $d(x,z)\le d(x,y)+d(y,z)$. (triangle inequality)

Here, we introduce a notion of distance over a given solution space which meets above three properties.

Definition 3 (Distance) Given a solution space (S,o), a distance of two solutions $s_i\in S$ and $s_j\in S$ is the least number of consecutive applications of the operator required for transforming s_i into s_j. The definition implies that there exists a sequence of applications of the operator: $o(s_i)=s_k, o(s_k)=s_l, ..., o(s_t)=s_j$.

Under the above notion of distance, a given (discrete) solution space (S,o) can induce a corresponding metric space (M,d) where $M=S$ and the distance between two solutions $s_i\in S$ and $s_j\in S$, $d(s_i,s_j)$ conforms to definition 3. For the metric space (M,d) induced by the solution space (S,o), there is an intuitive geometric explanation for the distance $d(s_i,s_j)$ of s_i and s_j, i.e. it is the length of a shortest path between the two nodes, respectively representing s_i and s_j, in the associated state-space graph. In above example, there are two shortest paths from 000 to 101 with length 2, i.e. $000\leftrightarrow 001\leftrightarrow 101$ and $000\leftrightarrow 100\leftrightarrow 101$. So $d(000,101)=2$. Indeed, such a distance is the so-called Hamming distance.

Definition 4 (Line segment) Given a metric space (M,d), a line segment between $x\in M$ and $y\in M$ is a set $\{z\mid z\in M\wedge d(x,z)+d(z,y)=d(x,y)\}$, usually denoted as $[x;y]$. Here x and y are called extremes of the segment. The length of a line segment is equal to the distance between its two extremes.

For above example, line segment $[000;111]=\{000,001,100,101\}$.

Let (M,d) be the metric space induced by a solution space (S,o). From the geometric perspective, the line segment between x and y is the set of all solutions represented by vertices on any shortest paths connecting vertices representing x and y, respectively. The length of the line segment is the length of any shortest path.

B. Generalization of PSO to Discrete Space

In canonical PSO, the velocity v and position (candidate solution) x of a particle at time step $t+1$ are updated as

$$v(t+1)=wv(t)+c_1r_1(p(t)-x(t))+c_2r_2(g(t)-x(t)) \quad (1)$$

$$x(t+1)=x(t)+v(t+1) \quad (2)$$

where p and g are personal best position of the particle and the best position of its neighborhood, respectively. w is an inertia weight, c_1 and c_2 are two

acceleration coefficients, and r_1 and r_2 are two random numbers in $[0,1]$.

To generalize PSO to discrete solution space, we are required to redefine all the arithmetic operators in equation (1) and (2). Most important of them is the difference of positions of two particles. The basic idea of traditional PSO is any particle moves close to the best of its neighbors and return to the best position of itself so far, in other words, the particle try to reduce both the distance from the best of its neighbors and the distance from the best position of itself so far. In terms of the notion of distance over a (discrete) solution space, for any positions x and y, x can be transformed into y through applying the operator along one of shortest paths between them in turn. Every application of the operator decreases the distance between x and y by 1. Thus, the difference of two positions (candidate solutions) of particles can be viewed as any sequence of applications of operator which transforms one position into the other one.

Definition 5 (Difference of two positions) For any position x and y, the difference of them, $x-y$, is a sequence of least number of consecutive applications of operator i.e. $x-y=v=o(x),o(r),....,o(t)$ such that $x,r,s,...,t,y\in[x;y],o(x)=r,o(r)=s,....,o(t)=y$. Difference of two positions results in a velocity, in other words, a velocity is also a sequence of applications of operator.

Definition 6 (Product of number and velocity) Supposing that $\phi\in[0,1]$ is a real number and $v=o^1o^2...o^d$ ($o^i(i=1,2,...,d)$ is the i-th application of the operator) is a velocity, the product of them, ϕv is a subsequence of v such that $\phi v=o^1o^2...o^k(k=\lceil cd\rceil)$.

Definition 7 (Sum of position and velocity) Supposing that x is a position and $v=o^1o^2...o^d$ is a velocity, the sum of them, $x+v$, is a new position such that $x+v=o(...o(o(x)))$ (d consecutive applications of the operator).

In Eq. (1), the velocity of particle is composed of three components, known as inertia, cognitive component and social component, respectively. Since velocity is defined as a sequence of applications of operator which transforms a position into another position or iconically a shortest path from one position to another, sum of two velocities is deficient in meaningful geometric explanation. Therefore, the sum of two velocities is not introduced. Particles, however, are driven by two forces: one from its best neighbor (corresponding to social component) and the other from its best position so far (corresponding to cognitive component). Here the two forces are dealt as follows: each one is assigned with a probability, $prob_p$ for force from personal best and $prob_n$ for force from neighbor best position, respectively. $prob_p+prob_n=1$ is met, i.e., at any time step, the update of a particle is influenced by exactly one of the two components.

Now, we can describe the discrete PSO algorithm as follows:

Algorithm 1 Discrete PSO algorithm
Input: a problem with discrete domain
Output: a (near) optimal solution

Create and initialize an N-dimensional swarm, S;
REPEAT
  FOR each particle $i\in S$ DO
    IF $f(x_i)<f(p_i)$ THEN $p_i\leftarrow x_i$;
  ENDFOR
  FOR each particle $i\in S$ DO
    $g_i\leftarrow\arg\min_{p_j}\{f(p_j)\mid j\in Neighbor(i)\subset S\}$;
  ENDFOR
  FOR each particle $i\in S$ DO
    Generate a random number $\delta\in[0,1)$
    IF $0\le\delta<prob_p$ THEN $v_i\leftarrow r_1(p_i-x_i)$;
    IF $prob_n\le\delta<1$ THEN $v_i\leftarrow r_2(g_i-x_i)$;
    $x_i\leftarrow x_i+v_i$;
  ENDFOR
UNTIL stopping condition is true;

Notice that the inertia component is omitted and the functionality of acceleration coefficients c_1 and c_2 which weight the contribution of personal best and neighborhood best to velocity are replaced by $prob_p$ and $prob_n$.

III. HYBRID DISCRETE PSO ALGORITHM FOR GCP

The usual definitions about graph coloring problem are formally formulated as follows:

Definition 8 (Graph k-coloring problem) Given a graph $G=(V,E)$, where V and E are respectively the set of vertices and the set of edges and a positive integer k, the graph k-coloring problem is to determine whether there exists a conflict-free vertex coloring using k colors or less, i.e. a function $c:V\rightarrow\{1,2,...,k\}$ such that $\forall(u,v)\in E,c(u)\neq c(v)$. If such a coloring exists, G is said to be k-colorable.

Definition 9 (Graph Coloring problem) Given a graph G, the graph coloring problem is to determine the smallest k such that G is k-colorable, i.e. to find out its chromatic number.

The above definitions view GCP as an assignment of colors to the vertices of the graph, which is not proper for optimization algorithms to solve the coloring problem because of permutation symmetry. In terms of Definition 8, for a given graph, the number of all possible k-colorings is $k^{|V|}$ (the number of all possible functions from V into $\{1,2,...,k\}$), in other words, there are $k^{|V|}$ candidate solutions for an optimization algorithm to search. Indeed, for any k-coloring $c:V\rightarrow\{1,2,...,k\}$ of a

graph, another coloring $c':V\to\{1,2,...,k\}$ such that $\forall v\in V, c'(v)=I(c(v))$ where $I:\{1,2,...,k\}\to\{1,2,...,k\}$ is a bijection (permutation) is the same k-coloring of the graph. A more efficient method involves the equivalence of GCP to the problem of partitioning the vertex set into k or fewer independent sets. In the sense of partition, coloring c and c' are identical. Under the notion, a coloring of a graph can be formulated as follows:

Definition 10 (Coloring) Given a graph $G=(V,E)$, a coloring C of G is a partition of V into disjoint subsets, i.e., $C=\{V_i \mid \bigcup V_i = V \wedge \forall i,j, i\neq j, V_i\cap V_j=\Phi\}$.

The number of possible partitions of V into k nonempty disjoint subsets is the Stirling number of the second kind $S(|V|,k)$. Hence, the number of all possible partitions of V into k or less disjoint subsets is $\sum_{i=1}^{k} S(|V|,i)$ which is much less than $k^{|V|}$. For this reason, to solve the graph k-coloring problem, an optimization algorithm may only search much less solutions.

The graph coloring problem can be viewed an optimization problem (to minimize k), while k-coloring problem is its corresponding decision problem (to determine whether there exists a k-coloring or not). If one can solve k-coloring problem, one can also solve coloring problem by the following iterative approach: find a k-coloring for a fixed k, then decrease k ($k=k-1$) until no conflict-free k-coloring can be found.

To solve a problem by above discrete PSO algorithm, the difference of two positions must be evaluated, i.e. find out a sequence of least number of applications of operator over solution set. Firstly, however, the solution space need to be determined, i.e. the set solution is determined and an operator is defined or designed.

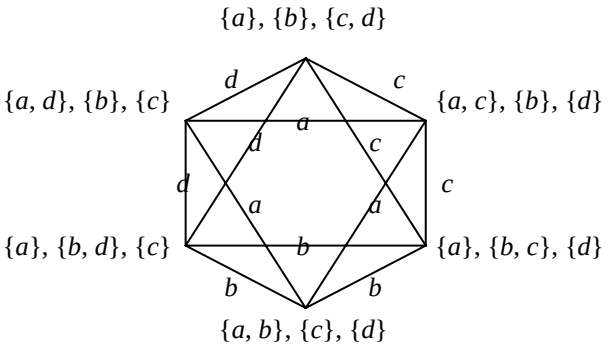

Fig.2 An example of state-space graph for solution space of 3-coloring problem

For k-coloring problem, an operator, say *move*, that transforms a solution into another solution is designed as changing the color of a vertex into another color or imaginatively moving a vertex from a color class into another color class. A constraint on the operator is that it can not be applied to a color class containing only one element to prevent generating an illegal solution. In Fig. 2, an example of state-space graph for solution space of 3-coloring problem is illustrated. The symbol on an edge is the vertex being moved for transforming into each other of two solutions connected by the edge.

Secondly, we need to calculate difference of two positions (candidate solutions), in other words, to find out the shortest path between two solutions. For graph coloring problem, we adopt the partition-based representation of solution formulated in Definition 10. In terms of [18], the problem of solving distance of two partitions can be reduced to an assignment problem. It's well known that an assignment problem can be solved by Hungarian algorithm with $O(|V|^3)$ time complexity [19]. By virtue of Hungarian algorithm, we can easily complete the construction of difference of two partitions. The full procedure is depicted as Algorithm 2. The cost matrix of corresponding assignment problem is constructed first. Then the maximum assignment problem is solved using Hungarian algorithm which generates maximum assignment indexed by the subscripts of selected elements. For k-coloring problem, this means that the vertices belonging to corresponding intersection of two color classes need not be moved. Finally, all other vertices must be moved for transforming one coloring into another. For details of Hungarian algorithm, see [19].

Algorithm 2 Calculation of difference of two partitions
Input: two partitions (colorings) of k-coloring problem $C^s=\{V_1^s,V_2^s,...,V_k^s\}$ and $C^t=\{V_1^t,V_2^t,...,V_k^t\}$
Output: a set Γ of vertices which will moved according to the sequence of applications of operator *move* transforming C^s into C^t

Create cost matrix $A_{k\times k}=(a_{ij})$;
FOR $i=1,2,...,k$ DO
  FOR $j=1,2,...,k$ DO
    $a_{ij}\leftarrow|V_i^s\cap V_j^t|$;
  ENDFOR
ENDFOR
Calculate maximum assignment $\Lambda=\{(i,j)\}$ using Hungarian algorithm;
$\Gamma\leftarrow\{\}$;
FOR any $(i,j)\notin\Lambda$ DO
  $\Gamma\leftarrow\Gamma\cup(V_i^s\cap V_j^t)$;
ENDFOR

Metaheuristic hybrids that in some way manage to combine the advantage of population-based methods with the strength of trajectory methods are often very successful [20]. For the discrete PSO (metaheuristic) for graph coloring problem, a local search (trajectory method) algorithm is inserted after each update of each particle. One of the successful local search methods that has been proposed for solving graph coloring problems is Tabucol [21]. Tabucol is a tabu search algorithm developed by Hertz and de Werra in 1986 [4]. Here, we use the improved version of Tabucol by Galinier and Hao [22].

Tabucol iteratively modifies the color of a single vertex to decrease progressively the number of conflicting edges until a legal k-coloring is obtained. A tabu list is used in order to escape from local optima and to avoid short term cycling. Let $move(v,i)$ denote moving vertex v to color class V_i and $\delta(v,i)$ denote the decrement of number of conflicts after execution of the movement. We describe the algorithm formally as follows:

Algorithm 3 Tabu search for k-coloring problem
Input: A graph $G=(V,E)$ and an initial k-coloring $C=\{V_1,V_2,....,V_k\}$ of G
Output: An improved k-coloring $C^*=\{V_1^*,V_1^*,....,V_k^*\}$

$C^* \leftarrow C$;
$iter \leftarrow 0$;
REPEAT
  Choose a candidate $move(v,i)$ with maximum $\delta(v,i)$;
  Add $move(v,i)$ to the tabu list for t iterations;
  Execute $move(v,i)$ and generate a new coloring C';
  IF $f(C')<f(C^*)$ THEN $C^* \leftarrow C'$;
  $iter \leftarrow iter+1$;
UNTIL $iter=MAX_{TABU}$ or $f(C^*)=0$

Notice that only critical moves involving conflicting vertices are considered. In [22], the tabu tenure t for a move depends on the number $F(C)$ of conflicting vertices in the current coloring: $t=U[0\text{-}9]+0.6F(C)$, where $U[0\text{-}9]$ is random integer between 0 and 9. The fitness function $f(C)$ for GCP evaluates the number of conflicting edges in the coloring C. MAX_{TABU} is the maximum number of iterations for Tabucol.

To mitigate the effect of premature convergence, a stochastic factor is introduced as one of components of velocity. The stochastic factor is stemmed from the influence of a point chosen at random imposing on the current velocity. For graph coloring problem, a coloring C^r is generated at random with probability $prob_r$ and then $r_3(C^r-C)$ serves as the impact of a stochastic coloring. Now, we present the hybrid discrete PSO algorithm for k-coloring problem in algorithm 4.

Algorithm 4 Hybrid Discrete PSO algorithm for k-coloring problem
Input: A graph $G=(V,E)$ and number of colors k
Output: A best k-coloring of G

Create and initialize a collection of N k-colorings, S;
$iter \leftarrow 0$;
REPEAT
  FOR each k-coloring $C \in S$ DO
    IF $f(C)<f(C^p)$ THEN $C^p \leftarrow C$;
  ENDFOR
  FOR each k-coloring $C \in S$ DO
    $C^g \leftarrow \arg\min_{C'}\{f(C') \mid C' \in Neighbor(C) \subset S\}$;
  ENDFOR
  FOR each k-coloring $C \in S$ DO
    Generate a random number $\delta \in [0,1)$;
    IF $0 \le \delta < prob_p$ THEN $C \leftarrow C + r_1(C^p - C)$;
    IF $prob_p \le \delta < prob_p + prob_n$ THEN
      $C \leftarrow C + r_2(C^g - C)$;
    IF $prob_p + prob_n \le \delta < 1$ THEN
      $C \leftarrow C + r_3(C^r - C)$;
    Improve C using Tabucol
  ENDFOR
  $iter \leftarrow iter - 1$;
UNTIL $iter = MAX_{PSO}$ or $f(C^g)=0$

To generate a stochastic solution of k-coloring problem, it's terrible that vertices are classified into k disjoint sets one by one because the k disjoint sets will be more or less the same size. In order to construct an initial population with abundant diversity, we design the following algorithm to initialize a solution of k-coloring problem.

Algorithm 5 Initialize a solution of k-coloring problem
Input: A graph $G=(V,E)$ and number of colors k
Output: A random k-coloring $C=\{V_1,V_2,...,V_k\}$ of G

$\Gamma \leftarrow V$;
FOR $i=1,2,...,k$ DO
  $V_i \leftarrow \{\}$;
ENDFOR
Generate randomly k positive integers $n_1,n_2,...,n_k$ such that $\sum_{i=1}^{k} n_i = |V|$;
FOR $i=1,2,...,k$ DO
  FOR $j=1,2,...,n_i$ DO
    Choose a vertex $v \in \Gamma$ at random, $V_i \leftarrow V_i \cup \{v\}$, $\Gamma \leftarrow \Gamma \setminus \{v\}$;
  ENDFOR
ENDFOR

IV. Experiment & Discussion

To demonstrate the performance of the proposed hybrid discrete PSO (HPSO for short), an extensive experiments on some benchmarks are conduct.

The demonstrated graphs are derived from the well-known second DIMACS challenge benchmarks [23], illustrated in Table 1. These instances cover a variety of types and sizes of graphs.

TABLE 1 BENCHMARK GRAPHS

Graph	number of vertices	number of edges	Chromatic Number	Description
DSJC250.5	250	31336	Unknown	Random graphs with edge density 0.5
DSJC500.5	500	125248	Unknown	
le450_15c	450	16680	15	Leighton graphs with guaranteed coloring size
le450_15d	450	16750	15	
le450_25c	450	17343	25	
le450_25d	450	17425	25	
flat300_26	300	21633	26	Quasi-random coloring problem
flat300_28	300	21695	28	

TABLE 2 PARAMETER SETTING

Parameter	Swarm size	$prob_p$	$prob_n$	$prob_r$	MAX_{PSO}	MAX_{TABU}
Value	10	0.4	0.5	0.1	16|*V*|	16|*V*|

The values of parameters in HPSO algorithm for the graph coloring problems are presented in Table 2. These values are selected based on some preliminary trials. $MAX_{PSO} = 16|V|$ means that HPSO algorithm is terminated when it finds out a legal k-coloring or reaches $16|V|$ iterations. Similarly, $MAX_{TABU} = 16|V|$ means that each run of Tabucol is terminated when it finds out a legal k-coloring or reaches $16|V|$ iterations.

For each instance in Table 1, 10 independent runs of algorithm are carried out. The experimental results are shown in Table 3. Column 2 records chromatic number ("?" stands for unknown chromatic number) and best known result. Column 3 indicates various values of k which corresponds to various k-coloring problems solved by HPSO. Column 4 reports the number of successful runs and the number of tries. The average number of HPSO iterations on successful runs is presented in Column 5. And final column records the average CPU time (in seconds) cost by successful runs.

We also compare HPSO with several other well-known algorithms including HCA (a hybrid evolutionary algorithm by Galinier and Hao in [22]), VNS (a variable neighborhood search algorithm by Avanthay *et al*. in [9]), VSS (a variable space search algorithm by Hertz et al. in [10]) and ANTCOL (a hybrid ant colony optimization algorithm by Dowsland and Thompson in [8]). For each algorithm, the smallest *k* with which a legal k-coloring was found is reported in Table 4. It is worth notice that this is only a rough comparison because of various experimental conditions.

The experimental results in Table 3 show that our hybrid discrete PSO algorithm for graph coloring problem is feasible and robust. HPSO is capable of finding out an optimal coloring for graphs le450_15c, le450_15d and flat300_26 with a very high probability (100%). For random graphs DSJC250.5 and DSJC500.5, HPSO has also found best known optimal colorings. For the more difficult instance, graphs le450_25c, le450_25d and flat300_28, HPSO found out approximate best-known colorings. Moreover, HPSO is expected to improve the solution quality for most instances with additional iterations since all runs of HPSO on these instances are all successful. From Table 4, we can observe that HPSO is competitive with other well-known algorithms.

TABLE 3 EXPERIMENTAL RESULTS

Graph	χ, k^*	*k*	Succ/run	PSO Iter	CPU time (s)
DSJC250.5	?, 28	28	10/10	145.3	69
DSJC500.5	?, 48	48	1/10	5831	20726
		49	4/10	4305	13112
		50	10/10	1602.5	3642
le450_15c	15, 15	15	10/10	49.6	52
le450_15d	15, 15	15	10/10	68.9	69
le450_25c	25, 15	26	10/10	868.6	1352
le450_25d	25, 15	26	10/10	526.7	712
flat300_26	26, 26	26	10/10	20.3	57
flat300_28	28, 28	31	10/10	412.8	327

TABLE 4 COMPARISON OF HPSO WITH OTHER ALGORITHMS

Graph	χ, k^*	HPSO	HCA	VNS	VSS	ANTCOL
DSJC250.5	?, 28	**28**	**28**	—	—	**28**
DSJC500.5	?, 48	**48**	**48**	49	**48**	49
le450_15c	15, 15	**15**	**15**	**15**	**15**	**15**
le450_15d	15, 15	**15**	—	**15**	**15**	—
le450_25c	25, 15	26	26	—	26	**25**
le450_25d	25, 15	26	—	—	26	—
flat300_26	26, 26	**26**	—	31	—	—
flat300_28	28, 28	31	31	31	**29**	31

In the experiment, for the sake of simplicity, the algorithm adopted the same values of parameters for all benchmark graphs. From the viewpoint of optimization, however, the landscapes of the fitness functions of these graphs are greatly different from each other, which results in significant difference in difficulty of solving them. For instance, estimated by experiments, there are millions of local optimum 15-colorings in the neighborhood of radius 2 of a legal 15-coloring for le450_15c. The neighborhood of radius 2 of a coloring is the set of all neighbors with distance 2 from the coloring. Formally, for a coloring C, its neighborhood of radius 2 is $N(C) = \{C' \mid d(C, C') = 2\}$. For more difficult instances, there are much more local optima in the whole solution space.

In the light of great difference of distinct graphs, it's possible to improve the solution of coloring of each graph by fine-tuning values of parameters for each graph.

V. Conclusion

In this paper, we attempted an adaptation of PSO to graph coloring problem. For generalization of PSO to discrete problem, we introduced a notion of distance over any discrete solution space. A distance is defined as the least number of consecutive applications of the operator on the solution space. The definition is a general concept provided a definite set of solutions and an operator on solution is given. Under this notion of distance, we redefined the standard PSO operators based on the basic idea of PSO. After reinterpreting the composition of velocity of a particle, we proposed a framework of PSO algorithm for any discrete problem.

The key to apply the discrete PSO algorithm to solve discrete problem is calculate difference of two positions. For graph coloring problem, we determined its solution space and then designed an algorithm to evaluate difference of two colorings, i.e. to find out the sequence of applications of operator transforming one coloring into the other one. Combining a local search, we proposed a hybrid discrete PSO algorithm for graph coloring problem, named HPSO. Experiment on a set of eight DIMACS benchmarks was conducted and the computational results show that HPSO is feasible and competitive with other well-known algorithms.

For time limit, we just tested our algorithm on several instances. In the future, more tests will be carried out to check the performance of our algorithm. Moreover, there is still a big room for improvement of our algorithm in the future. Also, the proposed discrete PSO algorithm will be applied to other combinatorial optimization problems.

Acknowledgement

We would like to thank the referees for their thoughtful comments.

References

[1] M. R. Garey & D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness. W.H. Freeman, New York, 1997.

[2] D. Brélaz, New methods to color vertices of a graph, Communications of ACM, vol. 22, pp. 251–256, 1979.

[3] F.T. Leighton, A graph coloring algorithm for large scheduling problems, Journal of Research of the National Bureau Standard, vol. 84, pp. 489–505, 1979.

[4] A. Hertz, D. de Werra: Using tabu search techniques for graph coloring. Computing 39(4), 345–351, 1987

[5] R. Dorne, J. K. Hao: Tabu search for graph coloring, T-colorings and set Tcolorings. Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization, 77–92, 1998

[6] D. S. Johnson, C. R. Aragon, L. A. McGeoch, C. Schevon: Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning. Operations Research 39(3), 378–406, 1991

[7] C. Fleurent, J. A. Ferland: Genetic and hybrid algorithms for graph coloring. Annals of Operations Research 63, 437–461, 1996

[8] K. A. Dowsland and J. M. Thompson, An improved ant colony optimisation heuristic for graph colouring, Discrete Applied Mathematics, Volume 156, Issue 3, Pages 313-324, 2008

[9] C. Avanthay, A. Hertz, N. Zufferey, A variable

neighborhood search for graph coloring, European Journal of Operational Research 151, 379–388, 2003

[10] A. Hertz, M. Plumettaz, N. Zufferey, Variable space search for graph coloring, Discrete Applied Mathematics 156, 2551–2560, 2008

[11] J. Kennedy & R. C. Eberhart, Particle Swarm Optimization, Proceedings of the 1995 IEEE International Conference on Neural Networks (Perth, Australia): IEEE Service Center, Piscataway, NJ, IV: pp 1942-1948, 1995.

[12] R. Poli, Analysis of the Publications on the Applications of Particle Swarm Optimisation, Journal of Artificial Evolution and Applications, 2008.

[13] A. P. Engelbrecht, Fundamentals of computational swarm intelligence. Chichester: Wiley. 2005

[14] J. Kennedy & R. C. Eberhart, A discrete binary version of the particle swarm algorithm. In Proceedings of the conference on systems, man, and cybernetics pp. 4104–4109. Piscataway: IEEE, 1997.

[15] M. Clerc, Discrete particle swarm optimization, illustrated by the traveling salesman problem. In B. V. Babu & G. C. Onwubolu (Eds.), New optimization techniques in engineering (pp. 219–239). Berlin: Springer, 2004.

[16] A. Moraglio, C. Di Chio, & R. Poli, Geometric particle swarm optimization. In M. Ebner et al. (Eds.), Lecture notes in computer science: Vol. 4445. Proceedings of the European conference on genetic programming (EuroGP). (pp. 125–136). Berlin: Springer, 2007.

[17] T. Gong & A. L. Tuson, "Forma Analysis of Particle Swarm Optimisation for Permutation Problems," Journal of Artificial Evolution and Applications, Volume 2008, Article ID 587309, 16 pages, 2008.

[18] D. Gusfield, Partition-distance: A problem and class of perfect graphs arising in clustering, Information Processing Letters, Volume 82, Issue 3, 16 May 2002, Pages 159-164

[19] R. Burkard, M. Dell'Amico, S. Martello, Assignment Problems, Society for Industrial and Applied Mathematics, Philadelphia, 2009

[20] C. Blum, A. Roli. Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison. ACM Computing Surveys (CSUR), Volume 35, Issue 3, September 2003. Pages: 268 - 308

[21] P. Galiniera, A. Hertz, A survey of local search methods for graph coloring, Computers & Operations Research 33, 2547–2562, 2006

[22] P. Galinier, J. K. Hao: Hybrid evolutionary algorithms for graph coloring. Journal of Combinatorial Optimization 1999; 3: 379–97

[23] http://mat.gsia.cmu.edu/COLOR/instances.html.