Dynamic Inertia Weight Particle Swarm Optimization for Solving Nonogram Puzzles

Habes Alkhraisat
Department of computer science
Al-Balqa Applied University
AL salt, Jordan

Hasan Rashaideh
Department of computer science
Al-Balqa Applied University
Al Salt, Jordan

***Abstract*—Particle swarm optimization (PSO) has shown to be a robust and efficient optimization algorithm therefore PSO has received increased attention in many research fields. This paper demonstrates the feasibility of applying the Dynamic Inertia Weight Particle Swarm Optimization to solve a Non-Polynomial (NP) Complete puzzle. This paper presents a new approach to solve the Nonograms Puzzle using Dynamic Inertia Weight Particle Swarm Optimization (DIW-PSO). We propose the DIW-PSO to optimize a problem of finding a solution for Nonograms Puzzle. The experimental results demonstrate the suitability of DIW-PSO approach for solving Nonograms puzzles. The outcome results show that the proposed DIW-PSO approach is a good promising DIW-PSO for NP-Complete puzzles.**

Keywords—Non-Polynomial Complete problem; Nonograms puzzle; Swarm theory; Particle swarms; Optimization; Dynamic Inertia Weigh

I. Introduction

Most of optimization problems including NP-complete problem, such as Nonograms puzzle, have complex characteristics with heavy constraints. Nonograms are deceptively simple logic puzzles, which is considered as an image reconstruction problem, starting with a blank N × M grid, Fig. 1.a shows an example for 5 x 5 Nonograms puzzle.

The solution of the puzzle is an image grid that satisfies certain row and column constraints. The constraints take the form of series of numbers at the head of each line (row or column) indicating the size of blocks of contiguous filled cells found on that line.

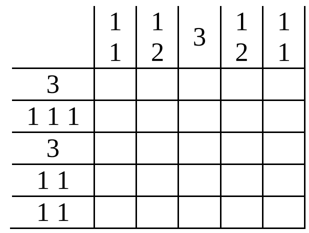
The puzzle solvers need to figure out which square will be left blank (white) and which will be colored (black), based on the numbers at the side of the grid. The resulting pattern of colored or left blank squares makes up a hidden picture, which is the solution to the puzzle.

The resulting picture must obey all the following three conditions:

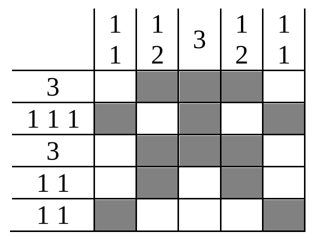
1) Each picture cell must be either colored or blanked i.e. black or white.

2) The $s_1, s_2, \ldots, s_k$ numbers at the side of the row or column: indicated that there are groups of s_1, s_2, and s_k filled squares, with at least one blank square between consecutive groups.

3) Between two consecutive black there must be at least one empty cell.



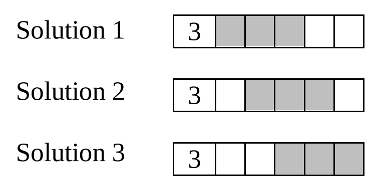
(a) 5 × 5 Nonograms puzzle



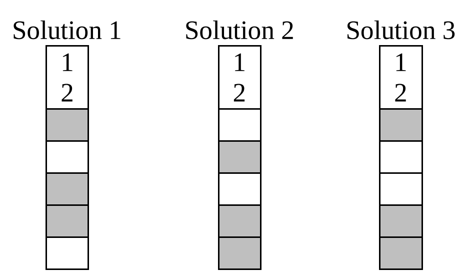
(b) 5 × 5 Nonograms solution

Fig. 1. (a) 5 × 5 Nonograms puzzle (b) its solution

For example, in the first row the "3" tells that, somewhere in the row, there are three sequential blocks filled in. Those will be the only blocks filled in, and the amount of space before/after them are not defined. The possible solution for the first row are:



The "1 2" in the second columns tells that, somewhere in the column, there is one block filled in, followed by 2 sequential blocks filled in, and also those will be the only blocks filled in, and the amount of space before/after them are not defined. The possible solutions for the second column are:



A puzzle is complete when all rows and columns are filled, and meet their definitions, without any contradictions. Fig. 1 shows an example of a Nonograms and its solution.

Several algorithms have applied to find a solution for the Nonograms problem such as an evolutionary algorithm, a heuristic algorithm, and a reasoning framework [2, 3, 4, and 5].

In this paper, a Dynamic Inertia Weight Particle Swarm Optimization (DIW-PSO) algorithm is proposed for solving Nonograms puzzles. In this work, we demonstrate that DIW-PSO can be specified to NP-Complete puzzle.

II. Dynamic Inertia Weight Particle Swarm Optimization

Particle swarm optimization (PSO) is a population based stochastic optimization method, which is an efficient and effective global optimizer in the discrete search domain [6]. PSO has been successfully applied to a wide variety of problems in mechanical engineering, communication, pattern recognition and diverse fields of science.

In PSO, a multiple random candidate solutions, so-called particles, are maintain in the problem search space, where each particle represents a solution to an optimization problem. Each particle is assessed by fitness function to figure out whether a particle is the problem "best" solution or not. A particle then fly through the problem search space with a randomized velocity by combining the current and best potential solution locations.

Let D be the size of the swarm, each particle i is composed of the following D-dimensional vectors: (1) the current position $\overrightarrow{x_i}$, (2) velocity $\overrightarrow{v_i}$, and (3) best value $\overrightarrow{p_i}$.

The PSO algorithm consists of adjusting the velocity and position of each particle toward new current best and global best locations. At each time step, current position $\overrightarrow{x_i}$ is updated by velocity and evaluated as a problem solution, in case the particle finds a pattern that is better than any it has found previously, it is recorded in the vector $\overrightarrow{p_i}$. And also the best fitness result value is recorded in $Pbest_i$, for comparison on the next iterations. The PSO keeps finding better positions and updating both $\overrightarrow{p_i}$ and pbest$_i$.

Position of individual particles x_i at $k+1$ iteration is modified according to the following [7]:

$$x_i^{k+1} = x_i^k + v_{ik+1}^{k+1} \tag{1}$$

The particle position is adjusted using the particle velocity which is calculated using the following equation [8, 9]:

$$v_i^{k+1} = w \times v_i^k + c_1 \times r_1\left(Pbest_i^k - x_i^k\right) + c_2 \times r_2\left(Gbest^k - x_i^k\right) \tag{2}$$

where,

- $i = 1, 2, \cdots, n;$
- k : iteration index,
- v_i^k, and x_i^k: velocity and position of particle i at iteration k,
- $Pbest_i^k$: best position of particle i at iteration k
- $Gbest^k$: global best position in the whole swarm until iteration k,
- c_1: cognitive parameter coefficient,
- c_2: social parameter coefficient,
- r_1 and r_2: predefined random values in rang [0, 1],
- ω : inertia weight factor controlling the dynamics of flying,
- n: number of particles in the group

The inertia weight factor dynamically adjusts the velocity of particle and therefore it controls the exploration and exploitation of the search space. The nonlinearly decreasing inertia weight w is set as follow [10]:

$$\omega = \omega_{min} + \left(\frac{iter_{max} - iter}{Iter_{max}}\right)^n \times (\omega_{max} - \omega_{min}) \tag{3}$$

where,

- ω_{min}, and ω_{max}: lower and upper limit value of inertia weights,
- $Iter_{max}$: maximum number of iteration,
- $Iter$: current iteration,

In each iteration, ω inertia weight will decrease nonlinearly from ω_{max} to ω_{min} and n is the nonlinear modulation index.

Fig. 2 illustrates PSO search mechanism according to "(1)" and "(2)".

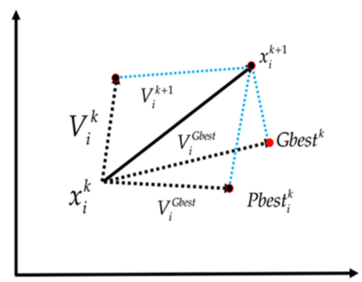


Fig. 2. The search mechanism of the particle swarm optimization

The process of PSO algorithm for solving Nonograms puzzles can be summarized as follows:

1) Initialization a population with random positions and velocities of a group of particles in d dimensional problem space while Nonograms puzzles constraints.

2) Position updating

3) Memory updating $Pbest$ and $Gbest$.

4) if stopping criteria is satisfied then stop PSO, else go to Step 2.

III. DIW-PSO for Solving Nonograms Puzzles

In this section, the DWI- PSO in solving Nonograms puzzle is described. The fitness function has a major role in the DWI-PSO algorithm, since it is the only standard of judging whether a particle is "best" or not. The fitness function for Nonograms puzzles is calculated as follow:

$$f_k^i(\mathrm{x}_\mathrm{k}^\mathrm{i}) = \sum_{p=0}^{n}\left|r_{i,p} - x_{ip}\right| + \sum_{p=0}^{m}\left|c_{p,i} - x_{p,i}\right| \tag{4}$$

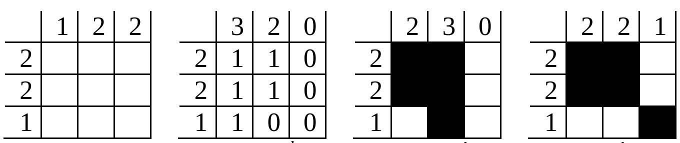
where

- $\chi_{i,r}$ is the total number of colored pixels at row r of individual i,
- Q_r is the total number of colored pixels at row r of the puzzle,
- $Y_{i,r}$ is the total number of colored pixels at column r of individual i,
- P_r is the total number of colored pixels at column r of the puzzle.

The fitness value $f_k^i(x_k^i)$ gives an indication how much the individual $\chi_{i,n}$ far from the optimal solution. Compare current particles fitness value $f(x_k^i)$ with best particles fitness value $f(Pbest_i^k)$. If $f(x_k^i)$ is better than $f(Pbest_i^k)$ then set f_{best}^i value to $f_k^i(x_k^i)$ and the $Pbes_k^i$ location to the location x_k^i. Then compare $f(x_k^i)$ with the population's global best $f(Gbest^k)$. If the $f(x_k^i)$ is better than $f(Gbest^k)$ then reset f_{best}^g to the current particle $f(x_i^k)$, and the $Gbest^k$ location to the location x_k^i . To illustrate the fitness function, consider the figure 2. The fitness function for figure 2 (b), (c) and (d) is calculated as follow:

$$f(Pbest_i^k) = |2-2| + |2-2| + |1-1| + |2-1| + |3-2| + |0-2| = 4$$
$$f(x_i^k) = |2-2| + |2-2| + |1-1| + |2-1| + |2-2| + |1-2| = 2$$
$$f(Gbest^k) = |2-2| + |2-2| + |1-1| + |3-1| + |2-2| + |2-0| = 4$$

Since $(x_i^k) < f(Pbest_i^k)$, the current x_i^k is better than $Pbest_i^k$, then set $f_{best}^i = 2$, and $Pbes_k^i = x_k^i$. And also since the $f(x_i^k) < f(Gbest^k)$, which indicates that current x_i^k is better than $Gbest^k$, then set $f_{best}^g = f(x_i^k)$, and $Gbest^k = x_k^i$.



(a) Nonograms puzzle (b) $Gbest^k$ (c) $Pbest_i^k$ (d) x_i^k

Fig. 3. An example to illustrate the Nonograms fitness function

At each iteration step, velocities of all particles are modified using "(2)", so the velocity of particle i at iteration k (Fig. 3) according to "(1)" is:

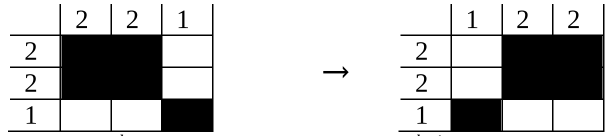
$$v_i^{k+1} = \lceil 1 \times 0 + 2 \times 0.2 \times (0) + 2 \times 0.8 \times (4) \rceil \; mod \; V_{max} = \lceil 6.4 \rceil \; mod \; 3 = 7 \; mode \; 3 = 1$$

where = 1 , $c_1 = c_2 = 2$, $v_i^k = 0$, $r_1 = 0.2$, $V_{max} = 3$, and $r_2 = 0.8$

After calculating the velocity, and between successive iterations, the modification of the particle position is controlled by the new calculated velocity. The modified position of x_i^k is done by adding the v_i^{k+1} to the x_i^k, as defined in "(2)":

$$x_i^{k+1} = x_i^k + 1$$

The result of the above equation means that the current particle x_i^k must be shifted one cell to right. Fig. 4 illustrates the result of shifting x_i^k.



(a) x_i^k (b) x_i^{k+1}

Fig. 4. Particle Position modification

Generally, the procedure for the proposed algorithm consists of the following steps:

Step 1: Initialization

1.1. Constant variables c_1, c_2 and k_{max}.

1.2. Positions of a group of particles x_i^k.

1.3. Velocities of a group of particles v_i^k.

Step 2: Optimization

2.1. For each particle, evaluate fitness f_k^i using (4).

2.2. Compare the fitness of each individual with each $Pbest_i$.

If $f_k^i \le f_{best}^i$, then the new position of i^{th} particle is better than $Pbest_i$, then set $f_{best}^i = f_k^i$, $Pbes_k^i = x_k^i$

2.3. Compare the fitness of each individual with $Gbest^k$.

If $f_k^i \le f_{best}^g$, the new position of i^{th} particle is better than $Gbest^k$, then set $f_{best}^g = f_k^i$, $Gbest^k = x_k^i$

2.4. Calculate the inertia weight using (3).

2.5. Update all particle velocities according to (2).

2.6. Update all particle positions according to (1).

2.7. Increment k.

2.8. repeat steps 2.1 – 2.4 until a sufficient good fitness or a maximum number of iterations are reached.

Step 3: Terminate

DWI-PSO parameters are as in Table 1. To solve the Nonograms puzzle we set the population size equal to the number of rows times number of columns in the Nonograms puzzle, maximum Number of iterations are considered as 10, 20, 50,100 and 1000, respectively, $c1 = c2 = 2$, and $Var_{max}$and Var_{min} are equal to the length of the search space [6, 11]. In addition, the inertia weight starts with 1.4 and decreases nonlinearly to 0.4 [12].

TABLE I. PARAMETERS FOR DWI-PSO

Population Size (Swarm Size)	nPop	
Maximum Number of Iterations	$iter_{max}$	10, 20, 50, and 100
Intertia Coefficient	ω	1.0
Intertia Coefficient maximum value	ω_{max}	1.4
Intertia Coefficient minimum value	ω_{min}	0.4
Personal Acceleration Coefficient	c_1	2
Social Acceleration Coefficient	c_2	2
Decision Variables maximum value	Var_{max}	1
Decision Variables minimum value	Var_{min}	0

IV. EXPERIMENTAL RESULTS

To clarify the efficiency of the DIW-PSO algorithm on Nonograms puzzle, several experiments as carried out. The experiment involved three puzzles of each of the following difficulties: "5×5", "10×10", "15×15", "20×20", "25×25", "30×30", "35×35", "40×40", and 45×45. All puzzles were selected from http://www.nonograms.org.

Table 2 shows the success DIW-PSO in solving Nonograms puzzle. Success rate represents the number of runs out of the maximum number of iterations.

TABLE II. SUCCESS RATE OF VARIOUS METHODS

Problem size	**number of runs / maximum number of iterations**		
	Puzzle 1	**Puzzle 2**	**Puzzle 3**
5×5	5/10	6/10	8/10
10×10	45/50	40/50	30/50
15×15	44/50	32/50	34/50
20×20	89/100	70/100	77/100
25×25	85/100	87/100	94/100
30×30	200/1000	205/1000	194/1000
35×35	195/1000	222/1000	275/1000
40×40	215/1000	245/1000	320/1000
45×45	200/1000	250/1000	310/1000

V. CONCLUSION

In this paper, we presented a new algorithm for solving Nonograms. The process of PSO algorithm in finding optimal values follows the social behavior of bird flocks and fish schools which has no leader. Particle swarm optimization consists of a swarm of particles, where particle represent a potential solution. Particle will move through a multidimensional search space to find the best position in that space. Particle swarm optimization (PSO) is a promising scheme for solving NP-complete problems due to its fast convergence, fewer parameter settings and ability to fit dynamic environmental characteristics.

The Nonograms problem is known to be NP-hard. The challenge is to fill a grid with black and white pixels in such a way that a given description for each row and column, indicating the lengths of consecutive segments of black pixels, is adhered to.

Firstly, this paper investigates the principles Nonograms puzzle and the general procedure for finding the puzzle solution. Moreover, the principles and optimization steps of Dynamic Inertia Weight Particle Swarm Optimization DWI-PSO and the influence of different parameters on algorithm optimization has been introduced in details.

In this paper, DWI-PSO has been applied for solving Nonograms puzzle. A dynamic inertia weight introduced to increase the convergence speed and accuracy of the PSO while searching for the best solution from Nonograms puzzle. The excremental results demonstrate the effectiveness, efficiency and robustness of the proposed algorithms for solving large size Nonograms puzzles.

In summary, we presented a DWI -PSO algorithm that has been successfully applied to NP-Complete puzzles. For future work, we will consider DWI-PSO for more challenging NP-Complete puzzles such as the Cross Sum, Cryptarithms, and Corral Puzzle.

REFERENCES

[1] J. Kennedy, R. C. Eberhart and Y. Shi, Swarm Intelligence, Morgan Kaufmann Publishers, San Francisco, 2001.

[2] K.J. Batenburg and W. A. Kosters. A discrete tomography approach to Japanese puzzles. In Proceedings of the 16th Belgium-Netherlands Conference on Artificial Intelligence (BNAIC), pages 243-250, 2004.

[3] S. Salcedo-Sanz, E.G. Ortiz-Garcia, A.M. Perez-Bellido, J.A. Portilla-Figueras, and X. Yao. Solving Japanese puzzles with heuristics. In Proceedings IEEE Symposium on Computational Intelligence and Games (CIG), pages 224-231, 2007.

[4] K.J. Batenburg and W.A. Kosters. Solving Nonograms by combining relaxations. Pattern Recognition, 42:1672-1683, 2009.

[5] N. Ueda and T. Nagao. NP-completeness results for Nonogram via parsimonious reductions, preprint, 1996.

[6] J. Kennedy and R. Eberhart. Particle swarm optimization. Proceedings, IEEE International Conference on Neural Networks, 4:1942–1948, 1995.

[7] A. P. Engel Brecht, Fundamental of Computational Swarm Intelligent, First Ed. The atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England: John Wiley & Sons Ltd, 2005.

[8] Eberhart R C, Shi Y. (1998). Comparison between genetic algorithms and Particle Swarm Optimization. Porto V W,Saravanan N,Waagen D, et al.Evolutionary Programming VII.[S.l.]:Springer,1998:611-616.

[9] Eberhart R C, Shi Y. (2000). Comparing inertia weights and constriction factors in Particle Swarm Optimization. Proceedin gs of the Congress on Evolutionary Computating, 2000: 84-88.

[10] A. Chatterjee and P. Siarry, Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization, Computers and Operation Researches, vol.33, no.3, pp.859-871, 2006.

[11] J. Chen, Z. Qin, Y. Liu and J. Lu, Particle swarm optimization with local search, Proc. of the IEEE Int. Conf. Neural Networks and Brain, pp.481-484, 2005.

[12] Y. Shi and R. C. Eberhart, A modified particle swarm optimizer, Proc. of the IEEE Conf. Evolutionary Computation, pp.69-73, 1998.