Discrete Particle Swarm Optimization for Number Partitioning Problem

Motoyoshi Kurose Takahiro Imabeppu Satoshi Ono Shigeru Nakayama
Department of Information and Computer Science
Faculty of Engineering, Kagoshima University
Kagoshima-shi, 890-0065 Japan
{ot102004, sc102005, ono, shignaka}@ics.kagoshima-u.ac.jp

Abstract

This paper experimentally observes the behavior of Discrete Particle Swarm Optimization (DPSO), a variation of Particle Swarm Optimization (PSO) to deal with binary variables, in Number Partitioning Problem (NPP). PSO and DPSO are optimization techniques inspired by flocking birds. NPP is a combinatorial optimization problem with binary variables to partition a set of numbers into two sets, so that the sum of them in each set is as close as possible. Experiments have been performed with problems in which phase transition (a phenomenon of abrupt change of average complexity) was observed in previous works.
keywords: discrete particle swarm optimization, number partitioning problem, phase transition

1 Introduction

Particle Swarm Optimization (PSO) has been developed by Eberhart and Kennedy in 1995 [1], inspired by the movement of flocking birds. PSO updates each individual's position in the search space based on its velocity and some best found solutions in the past. In PSO, the position of each particle denotes a potential solution to the optimization problem. PSO has been introduced as an optimization technique in real-number spaces, where the trajectories are defined as changes in position on some dimension.

Discrete Particle Swarm Optimization (DPSO) is a modification of the PSO algorithm for solving problems with binary-valued solution elements [2]. In DPSO, a particle position is a binary value changing between 0 and 1. A particle velocity is a probability that its position takes on 0 or 1. Several other forms of discrete optimization have been explored using PSO other than DPSO [3, 4].

Number Partitioning Problem (NPP) is a famous combinatorial optimization problem whose variables are binary: given a set of numbers, partition it into two sets, such that the sum of the numbers in each set is as equal as possible [5]. NPP is NP-complete, and is contained in many scheduling applications.

This paper experimentally observes the behavior of DPSO in NPP. We focus on particle movements with varying inertia coefficient, and the search performance of DPSO with instances in which a phase transition was observed in previous works.

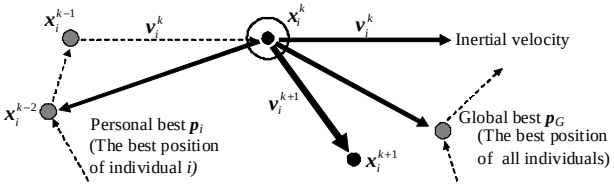


Figure 1: Calculation of velocity vector

2 Discrete Particle Swarm Optimization

PSO's optimizing process is quite simple. Each particle remembers the best previous position of it and the best previous position of all particles in the swarm. Each particle searches a good solution using two calculations. One is velocity update based on the present velocity of a particle, the best previous position of any particle and the best previous position of all particles in the swarm (Figure 1). Another is position update based on the present velocity.

$$\boldsymbol{v}_i^{k+1} = w \cdot \boldsymbol{v}_i^k + c_1 \cdot r_1 \cdot \left(\boldsymbol{p}_i - \boldsymbol{x}_i^k\right) + c_2 \cdot r_2 \cdot \left(\boldsymbol{p}_g - \boldsymbol{x}_i^k\right) \quad (1)$$

$$\boldsymbol{x}_i^{k+1} = \boldsymbol{x}_i^k + \boldsymbol{v}_i^{k+1} \quad (2)$$

where $\boldsymbol{v}_i$, $\boldsymbol{x}_i$, $\boldsymbol{p}_i$ are the velocity, position, and best previous positions of particle i at step k respectively, and $\boldsymbol{p}_g$ is the best previous position of all particles in the swarm. w is the inertia coefficient which slows the velocity over time to prevent explosions of the swarm and ensure ultimate convergence, c_1 is the weight given to the attraction to $\boldsymbol{p}_i$ and c_2 is the weight given to the attraction to $\boldsymbol{p}_g$. r_1 and r_2 are samplings of a uniformly-distributed random variable in $[0,1]$. Velocity $\left|v_{ij}^{k+1}\right|$ does not go beyond v_{max}, where j is the dimension of particle.

DPSO is a modification of the PSO algorithm for solving problems with binary-valued solution elements [2].

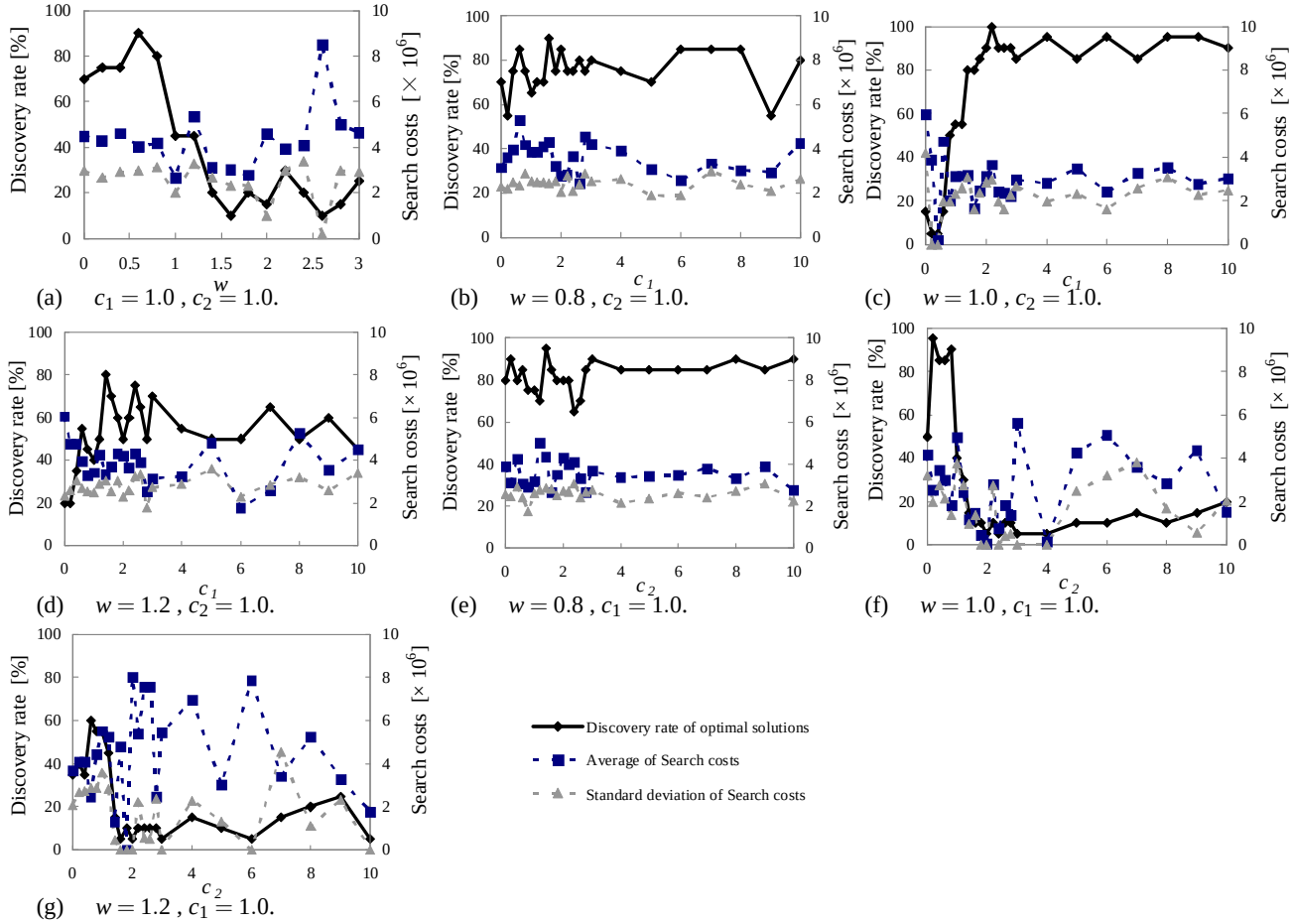


Figure 2: Experimental results on parameter w , c_1 , c_2 .

Similar to the optimization process of PSO, DPSO iterates velocity calculation and position update. However, a particle position in DPSO is expressed as a set of discrete value (0 or 1) to solve problems with binary-valued solution elements, while a particle position in PSO is expressed as a set of real value. In DPSO, each velocity value is transformed to the range (0, 1) by $s(v_{ij})$, and the particle position value is obtained by comparing this transformed element with a uniform random value.

$$x_{ij}^{k+1} = \begin{cases} 1 & if \quad (r() < s(v_{ij})) \\ 0 & otherwise \end{cases} \tag{3}$$

$$s(x) = \frac{1}{1+e^{-x}} \tag{4}$$

where (4) is the sigmoid function, and r is a value sampled randomly from 0.0 to 1.0.

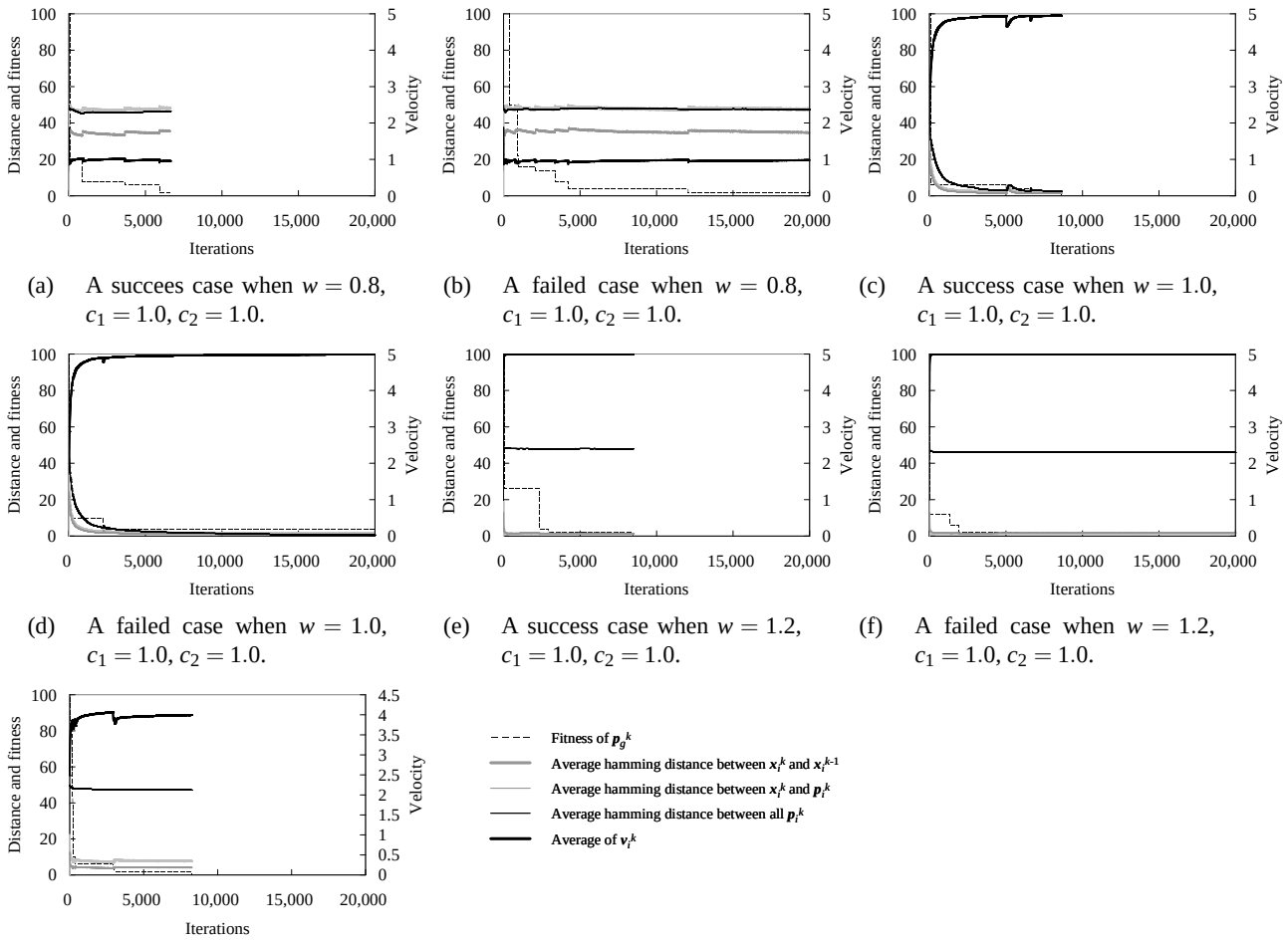
3 Number Partitioning Problem

NPP is a famous combinatorial optimization problem whose variables are binary: a finite set $A = \{a_k \mid k \in \{1 \ldots N\}\}$ of N positive integers, find a partition of A into two subset A_1 and A_2 such that difference

$$E(A) = \left| \sum_{i \in A_1} a_i - \sum_{i \in A_2} a_i \right| \tag{5}$$

is minimized [5]. A solution with $E = 0$ or $E = 1$ for $\sum A$ even or odd is called "perfect solution" for obvious reasons.

The phase transition in NPP is a phenomenon that there is a boundary that separates the "easy-to-solve" from the "hard-to-solve" phase. Phase transitions have been observed in many different NP-complete problems [7]. NPP's phase transition goes hand in hand with a transition in probability of perfect solutions [6]. Mertens [6] has showed that NPP has a phase transition in average complexity from experiments using complete differencing and complete greedy.



(a) A succees case when $w = 0.8$, $c_1 = 1.0$, $c_2 = 1.0$.

(b) A failed case when $w = 0.8$, $c_1 = 1.0$, $c_2 = 1.0$.

(c) A success case when $w = 1.0$, $c_1 = 1.0$, $c_2 = 1.0$.

(d) A failed case when $w = 1.0$, $c_1 = 1.0$, $c_2 = 1.0$.

(e) A success case when $w = 1.2$, $c_1 = 1.0$, $c_2 = 1.0$.

(f) A failed case when $w = 1.2$, $c_1 = 1.0$, $c_2 = 1.0$.

(g) A success case when $w = 1.0$, $c_1 = 2.2$, $c_2 = 0.2$.

Figure 3: Analytical results on varying w.

4 Computational Experiments

4.1 Pre-experimental planning

Section 4 evaluates the effectiveness of DPSO which is applied to NPP. In this paper, "search costs" is the fitness calculation time, which is calculated by *number of particle* $\times$ *iterations counts*. "Optimal solution" is a solution with minimal value of $E(A)$ calculated by (5) in each problem instance, which is not always a perfect solution. The number of particles, v_{max}, iteration limit were 500, 5.0, 20,000, respectively. We performed 20 runs for each condisiton.

4.2 Experiment 1

At first, we experimented with DPSO performance changing parameters w, c_1, and c_2 in Experiment 1. This experiment uses an instance involving $N = 100$ positive integer numbers less than 1,000,000.

First, we examined the effect of modifying parameter w between 0.0 and 3.0 with fixed $c_1 = 1.0$ and $c_2 = 1.0$. Figure 2(a) shows the result on discovery rate of optimal solutions and search costs. The graph shows a tendency that DPSO could find the optimal solution with higher rate when $w < 1.0$ than when $w > 1.0$.

Next, we examined the effects of parameters c_1 and c_2 changed between 0.0 and 10.0, in turn. w was set to 0.8, 1.0, and 1.2. Figure 2(b), (c), and (d) show the results on c_1, and Figure 2(e), (f), and (g) show the results on c_2.

From Figure 2(b) and (e), c_1 and c_2 did not seem to affect the search performance of DPSO when $w = 0.8$. In particular, the fact that the case with $c_2 = 0.0$ was not so worse than the cases with $c_2 > 0.0$ means that interaction between particles did not work properly when $w = 0.8$.

Figure 2(c), (d), (f), and (g) indicate that values of c_1 lower than 1.0 or values of c_2 higher than 1.0 causes dete-

riorations in search performance.

4.3 Experiment 2

We analyzed DPSO behaviors with varying w in Experiment 2. Parameters (w, c_1, c_2) are set to $(0.8, 1.0, 1.0)$, $(1.0, 1.0, 1.0)$, $(1.2, 1.0, 1.0)$, and $(1.0, 2.2, 0.2)$, which was is the best parameter set in Experiment 1. The same problem instance is used as in Experiment 1.

Figure 3 shows transitions of some criteria in a success and failed cases: average $E(\boldsymbol{p}_g^k)$, average hamming distance between $\boldsymbol{x}_i^k$ and $\boldsymbol{x}_i^{k-1}$ ($D(\boldsymbol{x}_i^k, \boldsymbol{x}_i^{k-1})$), average hamming distance between $\boldsymbol{x}_i^k$ and $\boldsymbol{p}_i^k$ ($D(\boldsymbol{x}_i^k, \boldsymbol{p}_i^k)$), average hamming distance between all $\boldsymbol{p}_i^k$ ($D(all\boldsymbol{p}_i^k)$), and average $\left|v_{ij}^k\right|$.

From Figure 3(a) and (b), $D(\boldsymbol{x}_i^k, \boldsymbol{x}_i^{k-1})$ and $D(all\boldsymbol{p}_i^k)$ were steady at around 50, which denotes that particles moved quite actively, rapidly and separately like random search when $w = 0.8$. Note that average $\left|v_{ij}^k\right|$ leveled out at about 1.0. In DPSO, lower v_{ij}^k means that a particle i moves fast, oppositely to the meaning of the word "velocity".

From Figure 3(e) and (f), $D(\boldsymbol{x}_i^k, \boldsymbol{x}_i^{k-1})$ and $D(all\boldsymbol{p}_i^k)$ rapidly converged at around 1.0 and 50 respectively, which denotes that each particle moved and converged at scattered places indifferently to each other when $w = 1.2$. Average $\left|v_{ij}^k\right|$ steeply grew and kept v_{max}.

From Figure 3(c) and (d), $D(\boldsymbol{x}_i^k, \boldsymbol{x}_i^{k-1})$ and $D(all\boldsymbol{p}_i^k)$ gradually decreased, which denotes that particles slightly slowed down and crowded into a solution. The best parameter set $(w, c_1, c_2) = (1.0, 0.2, 2.2)$ also allowed particles to move cooperatively, as shown in Figure 3(g).

4.4 Experiment 3

Finally, we tested DPSO with instances in which NPP's phase transition has been observed. According to the pervious work [6], instances whose numbers of elements N are from 8 to 120 were used. Element numbers of them were less than 2^{20}. 100 instances were generated for each number of elements.

Figure 4 shows the discovery rate of optimal solutions and the probability that a given instance has a perfect solution. Figure 5 shows the transitions of the discovery rate of optimal solution. As shown in these figures, the discovery rate drastically falls down at the same point $N = 24$ as in the previous work [6].

5 Conclusion

In this paper, we studied the performance of DPSO in NPP. Experiments have showed that the behavior of DPSO deeply depends on a inertia coerricient w, and the existence of a phase transition of DPSO. In future, we plan to examine other discretize methods for PSO.

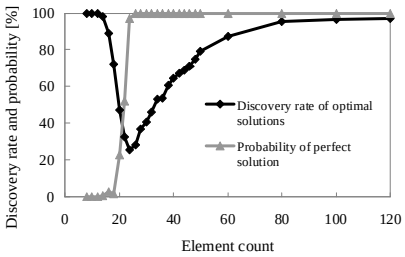


Figure 4: Discovery rate and probability

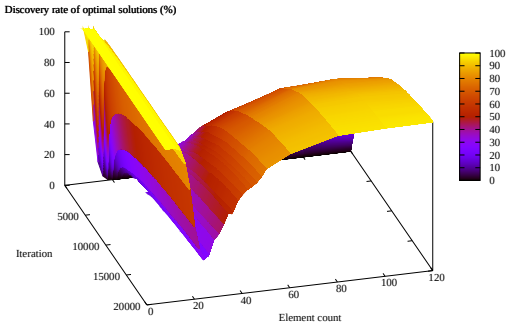


Figure 5: Discovery rate in DPSO

References

[1] J. Kennedy and R. C. Everhart, "Particle Swarm Optimization," *Proceedings of the 1995 IEEE International Conference on Neural Networks*, Vol. 4, pp. 1942-1948 IEEE Press, 1995.

[2] J. Kennedy and R. C. Everhart, "A discrete binary version of the particle swarm optimization algorithm," *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics 1997, Piscataway NJ*, pp. 4104-4109, 1997.

[3] Jim Pugh and Alcherio Martinoli, "Discrete Multi-Valued Particle Swarm Optimization," *Proceedings of IEEE Swarm Intelligence Symposium (2006)*, pp. 103-110, 2006.

[4] F. Afshinmanesh, A. Marandi and A. Rahimi-Kian, "A Novel Binary Particle Swarm Optimization Method Using Artificial Immune System," *IEEE International Conference on Computer as a Tool (EUROCON 2005), Serbia and Montenegro*, Vol. 1, pp. 217-220, 2005.

[5] W. Ruml, "Stochastic approximation algorithms for number partitioning," *Technical Report TR-17-93, Harvard University, Cambridge, MA, USA*, 1993.

[6] Stephan Mertens, "The Easiest Hard Problem : Number Partitioning," *Computational Complexity and Statistical Physics*, pp. 125-139, 2006.

[7] I.P. Gent and T. Walsh, "The Number Partition Phase Transition," *Proceedings of Workshop on Hard Problems, International Conference on Principles and Practice of Constraint Programming, Cassis, France*, 1995.