Dispersion and Velocity Indices for Observing Dynamic Behavior of Particle Swarm Optimization

The Jin Ai and Voratas Kachitvichyanukul

***Abstract*— A better balance of exploitation and exploration of solution space by the swarm is often mentioned as the key to a good performance of Particle Swarm Optimization (PSO) algorithm. Traditionally, the balance of exploitation and exploration ability of a PSO algorithm is usually shown empirically by the final result of the algorithm over some benchmark functions and not by the dynamic behavior of the swarm during the iteration process. In order to observe the dynamic behavior of the swarm in a PSO algorithm in details, two measurement indices, Dispersion Index and Velocity Index, are proposed. In an empirical study, these indices are embedded in two PSO Algorithms and applied to six benchmark problems. The results of this study indicate that a good balance between exploration and exploitation does lead to a better PSO. This balance could be achieved by allowing enough time or iteration step for both exploration and exploitation processes to take place. Finally, the utilization of these indices to balance strategy for exploitation and exploration on the PSO is discussed. It is also suggested that the velocity index can be used as a basis for controlling the length of iteration step of PSO algorithm.**

I. Introduction

PARTICLE Swarm Optimization (PSO) is a population based search method which were motivated by the group organism behavior such as bee swarm, fish school, and bird flock [1]. PSO imitated the physical movements of the individuals in the swarm as a search method, altogether with its cognitive and social behavior as local and global exploration abilities. In the PSO, a solution of a specific problem is being represented by an n-dimensional position of a particle. The search is performed by moving the particle to a new position via a velocity vector. The PSO algorithm is started with a population of particles initialized with random position and velocity. The population of particles is usually called a swarm. In one iteration step, every particle is moved from previous position to the new position based on its velocity. The velocity is updated based on the particle's personal best position and the global best position found so far by the swarm. Once a particle reach a position which has a better objective function than the best previous position for this particle, the personal best position is updated. Also, if an objective function is found that is better than the previous best objective function of the swarm, the global best position is updated. A brief and complete survey on PSO mechanism, technique, and applications is provided by [2] and [3].

Manuscript received June 15, 2007.

T. J. Ai is a doctoral candidate in Industrial Engineering and Management, School of Engineering and Technology, Asian Institute of Technology, Pathumtani 12120, THAILAND, on study leave from Universitas Atma Jaya Yogyakarta, INDONESIA (phone: 662-524-6604; fax: 662-524-5697; e-mail: st103468@ait.ac.th or jinai@mail.uajy.ac.id).

V. Kachitvichyanukul is an associate professor in Industrial Engineering and Management, School of Engineering and Technology, Asian Institute of Technology, Pathumtani 12120, THAILAND (e-mail: voratas@ait.ac.th).

Empirical study showed that PSO could be applied to solve unconstrained optimization [4], constrained optimization [5, 6], and also discrete optimization problems [7]. These early studies also showed that PSO could provide high quality solutions in reasonable fast computational time. Other studies were also carried out to deal with the issue of balancing exploration and exploitation ability of PSO. Exploration is the ability to test various regions in the problem space in order to locate good solutions, hopefully an optimal one. Exploitation is the ability to concentrate the search around a promising candidate solution in order to locate the optimum more precisely [8]. The results of these studies are the variants of PSO which were claimed to have better balance of exploration and exploitation ability than the original one [9, 10, 11] and parameter setting which enhances these ability of PSO [8, 12, 13].

In most of these studies, however, the improvement of exploration and exploitation ability was only demonstrated empirically by the final result obtained in solving some benchmark functions. The swarm behavior during the iterations has not been studied in details. This paper will fill this gap by studying the dynamics behavior of the swarm. Two measurement indices are proposed, dispersion and velocity index, for observing the swarm during the iterative process. These proposed indices are used to observe behavior of PSO algorithm on the benchmark problems. The remainder of this paper is organized as follow: Section 2 reviews the PSO algorithms and defines two measurement indices of the swarm. Section 3 shows the behavior of PSO algorithms on some benchmark problems with respect to proposed measurement indices. Section 4 discusses the findings of this study and directs the use of these indices to balance the exploitation and exploration strategies on the PSO. Finally, Section 5 summarizes this study and suggests further applications and extensions.

II. PSO Algorithms and Swarm Measurement Indices

Two versions of PSO Algorithm, the basic PSO and the GLNPSO, are briefly reviewed in this section. The usage of these two versions of PSO Algorithm is intended to show different exploration and exploitation behaviors that could be monitored by the two proposed swarm measurement indices.

In the basic PSO algorithm, each iteration step mainly consists of only two set of updating equations: velocity as in Equation (1) and position as in Equation (2).

$$v_{id} = wv_{id} + c_p u\left(p_{id} - x_{id}\right) + c_g u\left(p_{gd} - x_{id}\right) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

The velocity equation consists of three elements. First element shows that a particle, which is represented by its velocity, will maintain the current direction. Second element shows that it also uses its past knowledge to form a new direction which is shown in its cognitive behavior. Third element shows that it gains the information from other particles in the form of the best position of the swarm so far, and it showed the social behavior of particle. The PSO algorithm is described below following the definitions of the indices and notation.

Indices

i	:	index of particle, $i = 1 \ldots I$
d	:	index of dimension, $d = 1 \ldots D$
t	:	index of iteration, $t = 1 \ldots T$

Notation

X_i	:	the position vector of particle i, $X_i = \left[x_{i1},\ x_{i2},\ x_{i3},\ \cdots,\ x_{iD}\right]$
V_i	:	the velocity vector of particle i, $V_i = \left[v_{i1},\ v_{i2},\ v_{i3},\ \cdots,\ v_{iD}\right]$
P_i	:	the personal best position so far of particle i
P_g	:	the global best position so far of the swarm
$\phi(X_i)$	:	objective function value of particle i
$\phi(P_i)$	:	objective function of P_i, the best objective function of particle i
$v_{id}(t)$	:	the velocity of particle i at the dimension d in the iteration t
$x_{id}(t)$	:	the position of particle i at the dimension d in the iteration t
w	:	the inertia weight
c_p	:	the personal best acceleration constant
c_g	:	the global best acceleration constant
u	:	uniform random number in the range $[0,1]$
p_{id}	:	the personal best position of particle i at the dimension d
p_{gd}	:	the global best position at the dimension d

Algorithm 1: Basic PSO Algorithm

```
1. Initialize $I$ particles as a swarm population: generate the particle $i$ with random position $X_i$ in the range $\left[X^{\min}, X^{\max}\right]$, velocity $V_i = 0$ and personal best $P_i = X_i$ where $i = 1 \ldots I$. Set iteration $t = 1$.
2. For $i = 1 \ldots I$, compute the objective function of $X_i$, $\phi(X_i)$.
3. Update personal best: For $i = 1 \ldots I$, update $P_i = X_i$ if $\phi(X_i) < \phi(P_i)$.
4. Update global best: For $i = 1 \ldots I$, set $P_g = P_i$ if $\phi(P_i) < \phi(P_g)$.
5. Update velocity and position of each particle $i = 1 \ldots I$ and dimension $d = 1 \ldots D$:
   • $v_{id}(t+1) = wv_{id}(t) + c_p u\left(p_{id} - x_{id}(t)\right) + c_g u\left(p_{gd} - x_{id}(t)\right)$
   • $x_{id}(t+1) = x_{id}(t) + v_{id}(t+1)$
   • If $x_{id}(t+1) < X^{\min}$, then set $x_{id}(t+1) = X^{\min}$
   • If $x_{id}(t+1) > X^{\max}$, then set $x_{id}(t+1) = X^{\max}$
6. If the stop criterion is met, i.e. $t = T$, stop. Otherwise, set $t = t+1$ and return to step 2.
```

The GLNPSO is a PSO Algorithm with multiple social learning structures [11]. In this PSO version, the social behavior is expressed by not only the global best but also the local best and near neighbor best. The local best is the best position of among several adjacent particles. The near neighbor best is another social behavior concept, which is determined based on fitness-distance-ratio (FDR) [9]. The velocity updating equation is given in Equation (3).

$$\begin{aligned} v_{id} = {} & wv_{id} + c_p u\left(p_{id} - x_{id}\right) + c_g u\left(p_{gd} - x_{id}\right) \\ & + c_l u\left(p_{id}^{L} - x_{id}\right) + c_n u\left(p_{id}^{N} - x_{id}\right) \end{aligned} \quad (3)$$

where:

c_l	:	the local best acceleration constant
c_n	:	the near neighbor best acceleration constant
p_{id}^{L}	:	the local best position of particle i at the dimension d
p_{id}^{N}	:	the near neighbor best position of particle i at the dimension d

The personal and global best for GLNPSO are determined exactly as the basic PSO. The local best, P_i^L, is determined as the personal best with the least fitness value among K neighbors of particle i. Each dimension of the near neighbor best (p_{id}^N) is determined as the corresponding personal best (p_{jd}) that maximizing fitness-distance-ratio among all other particles. Where FDR is defined as

$$FDR = \frac{\varphi(X_i) - \varphi(P_j)}{\left|x_{id} - p_{jd}\right|} \text{ which } i \neq j \quad (4)$$

The GLNPSO Algorithm has the same structure as Algorithm 1. It has exactly the same step for all steps excluding step 4 and 5. In step 4, the GLNPSO contains procedure for updating global best, local best and near neighbor best. Equation 3 is applied as the updating velocity equation in step 5.

Algorithm 2: GLNPSO Algorithm

```
1. Initialize $I$ particles as a swarm population: generate the
   particle $i$ with random position $X_i$ in the range
   $\left[X^{\min}, X^{\max}\right]$, velocity $V_i = 0$ and personal best $P_i = X_i$
   where $i = 1 \ldots I$. Set iteration $t = 1$.
2. For $i = 1 \ldots I$, compute the objective function of $X_i$,
   $\phi(X_i)$.
3. Update personal best: For $i = 1 \ldots I$, update $P_i = X_i$ if
   $\phi(X_i) < \phi(P_i)$.
4. a. Update global best: For $i = 1 \ldots I$, set $P_g = P_i$ if
      $\phi(P_i) < \phi(P_g)$.
   b. Update local best: For $i = 1 \ldots I$, among all personal
      best from $K$ neighbors of the $i^{th}$ particle, set the
      personal best which obtains the least fitness value to be
      $P_i^L$.
   c. Generate near neighbor best: For $i = 1 \ldots I$, and
      $d = 1 \ldots D$, set $p_{id}^N = p_{jd}$ that maximizing fitness-
      distance-ratio ($FDR$) for $j = 1 \ldots I$.
5. Update velocity and position of each particle $i = 1 \ldots I$
   and dimension $d = 1 \ldots D$:
   • $v_{id}(t+1) = wv_{id}(t) + c_p u(p_{id} - x_{id}(t)) + c_g u(p_{gd} - x_{id}(t)) + c_l u(p_{id}^L - x_{id}(t)) + c_n u(p_{id}^N - x_{id}(t))$
   • $x_{id}(t+1) = x_{id}(t) + v_{id}(t+1)$
   • If $x_{id}(t+1) < X^{\min}$, then set $x_{id}(t+1) = X^{\min}$
   • If $x_{id}(t+1) > X^{\max}$, then set $x_{id}(t+1) = X^{\max}$
6. If the stop criterion is met, i.e. $t = T$, stop. Otherwise, set
   $t = t+1$ and return to step 2.
```

Two measurement indices are defined for observing the dynamic behavior of the swarm. The first index is called **dispersion index**. This index measures how particles are spreading around the best particle in the swarm, and is defined as the average absolute distance of each dimension from the best particle. The formula for the dispersion index ($\bar{\delta}$) is given in Equation 5. This index explains the coverage searching area of the swarm. A swarm with higher dispersion index has relatively wider coverage of searching area than the one with lower dispersion index.

$$\bar{\delta} = \frac{\sum_{i=1}^{I}\sum_{d=1}^{D}\left|x_{id} - p_{gd}\right|}{I \cdot D} \tag{5}$$

The second index is called **velocity index**. This index measures how fast the swarm moves in certain iteration, and is defined as the average of absolute velocity. The formula to calculate the velocity index ($\bar{\upsilon}$) is given in Equation 6. This index shows the moving behavior of the swarm: higher index means the swarm move more aggressively in the problem space than the swarm with lower index.

$$\bar{\upsilon} = \frac{\sum_{i=1}^{I}\sum_{d=1}^{D}\left|v_{id}\right|}{I \cdot D} \tag{6}$$

III. Computational Experiments

The computational experiment is performed using two PSO Algorithms: the basic version of PSO (Algorithm 1) and the GLNPSO (Algorithm 2). Parameters of the basic version of PSO algorithm for the experiments are $I = 30$, $T = 1000$, $c_p = 2$, $c_g = 2$, and w is linearly decreasing from 0.9 to 0.4. Parameters of the GLNPSO for the experiments are $I = 30$, $T = 1000$, $c_p = 1$, $c_g = 1$, $c_l = 1$, $c_n = 1$, $K = 5$, and w is linearly decreasing from 0.9 to 0.4. The measurement indices are coded and embedded in the PSO program, so that the PSO program can record or display the indices in every iteration step. Six benchmark functions, which are often used in PSO literature, are used for testing purpose. For all of these benchmark functions, the objective is to minimize it. The definition of each function is described below:

- Parabola 30D (search space: $[-20, 20]^{30}$)

$$\phi = \sum_{d=1}^{D} x_d^{\,2} \tag{7}$$

- Griewank 30D (search space: $[-300, 300]^{30}$)

$$\phi = 1 + \frac{\sum_{d=1}^{D}(x_d - 100)^2}{4000} - \prod_{d=1}^{D}\cos\left(\frac{x_d - 100}{\sqrt{d}}\right) \tag{8}$$

- Rosenbrock 30D (search space: $[-10, 10]^{30}$)

$$\phi = \sum_{d=1}^{D-1}\left[(1 - x_d)^2 + 100\left(x_d^2 - x_{d+1}\right)^2\right] \tag{9}$$

- Alpine 30D (search space: $[-10, 10]^{30}$)

$$\phi = \sum_{d=1}^{D}\left|x_d \sin(x_d) + 0.1 x_d\right| \tag{10}$$

- Ackley 30D (search space: $[-30, 30]^{30}$)

$$\phi = 20 - 20\exp\left(-0.2\sqrt{\sum_{d=1}^{D} x_d^{\,2} \Big/ D}\right) + e - \exp\left(\sum_{d=1}^{D}\cos(2\pi x_d) \Big/ D\right) \tag{11}$$

- Rastrigin 30D (search space: $[-10, 10]^{30}$)

$$\phi = \sum_{d=1}^{D}\left[x_d^{\,2} - 10\cos(2\pi x_d)\right] + 10D \tag{12}$$

From the equation 7–12, it is known that the search space for each problem is different. This relative size makes the comparison across functions difficult. To make the comparison convenience, the searching space in the PSO is always set in the interval $[0,1]$ instead of the range of original problem $\left[X^{\min}, X^{\max}\right]$ in this experiment. A linear translation from the PSO solution (i.e. x) to the original problem solution (i.e. x') is required by following relationship:

$$x' = X^{\min} + x\left(X^{\max} - X^{\min}\right) \tag{13}$$

The final objective function value (the global best objective function at the end of iteration) of each of these functions over five replications of both version of PSO are given in Table 1. Note that the optimal solutions for all these benchmark functions are zero. Hence, the GLNPSO version gives better results than the basic version of PSO for these benchmark functions, since it gives the final solutions that are closer to zero. This result is inline with the previous result [11]. However, observation of this final iteration result could only state empirically that the GLNPSO version is better than the basic one, but could not explain why this version is better.

To explain why one version of PSO is better, the dynamic of the swarm is studied by recording the dispersion and velocity index in every iteration step. Figures 1 and 2 shows the progress of dispersion index over one typical run of the basic version of PSO and the GLNPSO respectively, for the benchmark functions tested. For clarity of the figures, the data points are only shown for every 10 iterations for only three functions. Both figures show the general tendency of particle movements in the swarm: all particles move towards the global best position, so all particles are laid close to each other at the end of iteration.

It is also observed from Figures 1 and 2 that there is a different behavior of the swarm between the basic version of PSO and the GLNPSO. In the basic version of PSO, the swarm is shrinking slowly over iteration as observed by the dispersion index. It means that the coverage of searching area of the swarm is decreasing slowly over the iteration. Hence, the swarm could explore enough various regions of problem space. However, at the end of the iteration process the dispersion index is still far from zero or the swarm size is not yet small enough. This implied that there is enough time or iteration steps for exploration but not enough time for exploitation.

In the GLNPSO version, the swarm is shrinking more rapidly and approximately after the half step of iteration the dispersion index is nearly zero. At the first half of iteration, while the swarm size is big enough, the swarm could focus on exploring various regions in the problem space. Then, at the second half of iteration, since the swarm is clustered in a very small area, the swarm could be more concentrate to locate the optimum more precisely. It is implied that there is enough time for both exploration and exploitation processes in this version. Hence, it could be concluded that there is a good balance between exploration and exploitation. This balance may lead to a better solution than the basic PSO version.

TABLE I
COMPUTATIONAL EXPERIMENTS RESULT: COMPARISON OF FINAL OBJECTIVE FUNCTION VALUE

Function	Parabola		Griewank		Rosenbrock	
Replication	Basic PSO	GLNPSO	Basic PSO	GLNPSO	Basic PSO	GLNPSO
1	9.04E-05	3.02E-20	1.05E+01	7.39E-03	8.06E+01	2.42E+01
2	6.65E-05	2.01E-18	1.08E+01	2.47E-03	2.33E+01	2.42E+01
3	1.02E-03	2.39E-15	2.05E+01	1.48E-02	1.39E+02	8.42E+01
4	4.54E-05	3.57E-22	1.01E+01	1.72E-02	8.27E+01	2.20E+01
5	3.26E-04	2.13E-19	1.01E+01	9.85E-03	1.01E+04	2.71E+01
Function	Alpine		Ackley		Rastrigin	
Replication	Basic PSO	GLNPSO	Basic PSO	GLNPSO	Basic PSO	GLNPSO
1	8.88E+00	2.69E-03	1.78E+00	9.31E-01	6.28E+01	3.88E+01
2	4.45E+00	4.83E-04	2.18E-01	1.61E-10	9.45E+01	5.97E+01
3	2.18E-02	1.36E-03	1.25E-02	2.17E-07	5.91E+01	3.28E+01
4	8.91E+00	1.83E-04	9.31E-03	1.29E-09	4.08E+01	3.88E+01
5	2.14E-02	2.16E-05	1.37E+01	1.14E-08	6.32E+01	5.27E+01

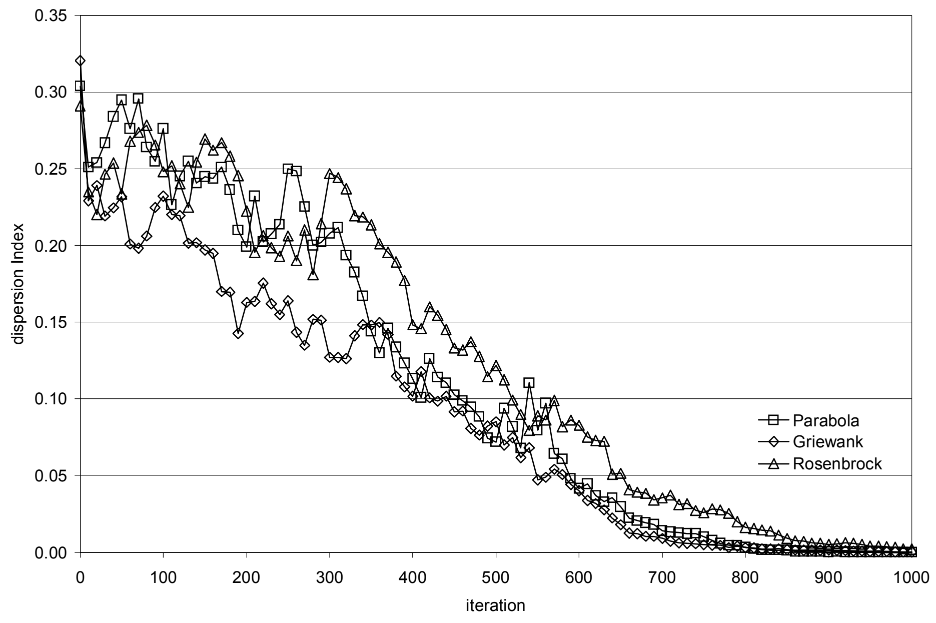

Fig. 1. Dispersion index on typical run of basic version PSO.

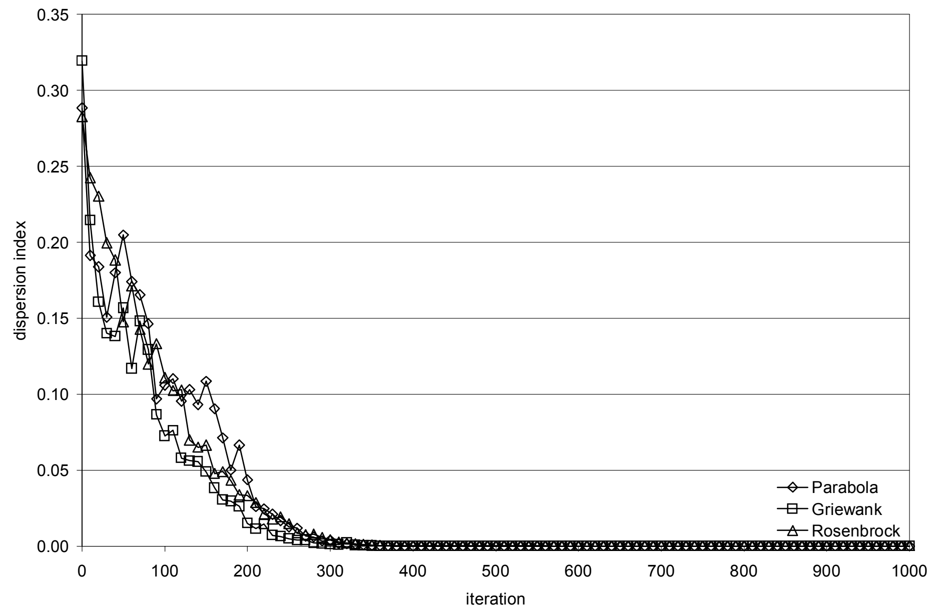

Fig. 2. Dispersion index on typical run of GLNPSO.

The progress of velocity index over iteration for the typical run of both PSO versions are presented in Figures 3 and 4. The pattern of velocity index in both versions are quite similar with theirs dispersion index. While the velocity index decreases slowly over iteration process for the basic PSO version; it diminishes rapidly for the GLNPSO version and the index become very small approximately after half of iteration process. It means that in the basic PSO version the swarm movement is decreasing slowly, but there is still significant movement at the end of iteration process. In other words, there is not enough iteration steps for exploitation. While in the GLNPSO version, particles in the swarm move aggressively in exploring problem space and move very slowly in exploiting the solution at the later iterations. Once again, this velocity pattern emphasizes the statement that balancing between exploitation and exploration may lead to better solutions.

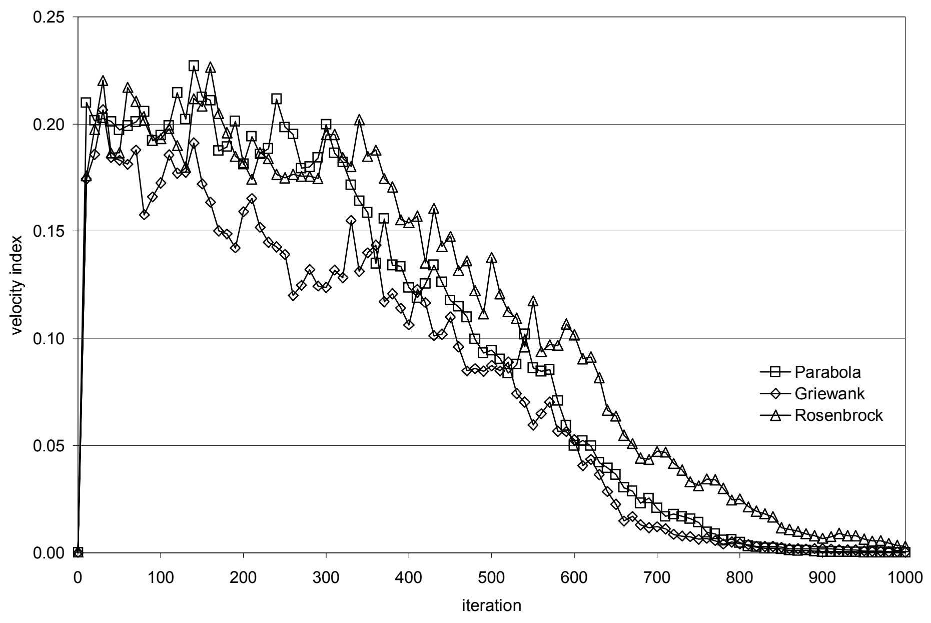

Fig. 3. Velocity index on typical run of basic version PSO.

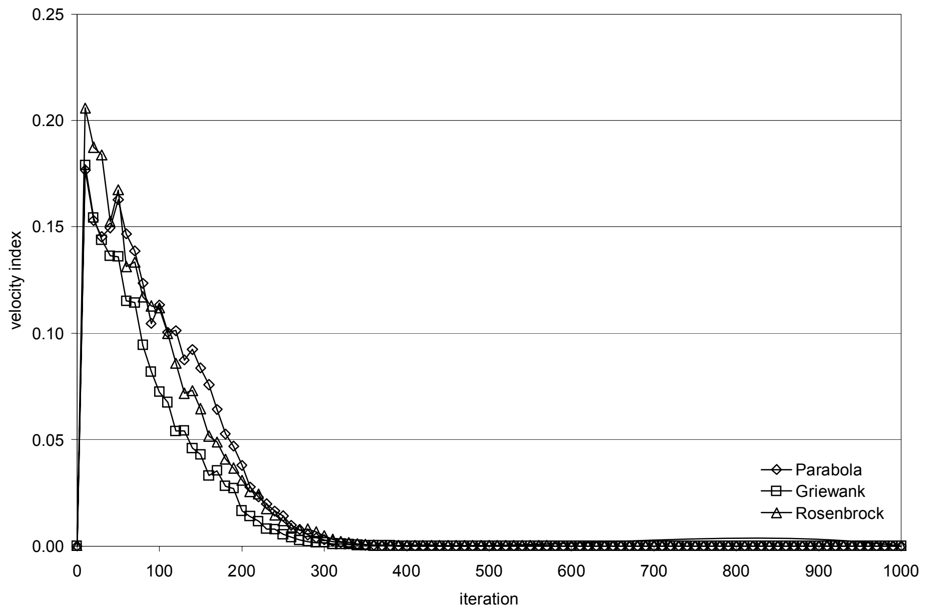

Fig. 4. Velocity index on typical run of GLNPSO.

IV. Controlling Exploration and Exploitation using Dispersion or Velocity Index

From the computational experiments in Section 3, it reiterated that a good balance between exploration and exploitation in a PSO algorithm will provide for better solution quality. It is also shown that the dispersion and velocity index could monitor when the shift from exploration to exploitation processes possibly takes place. Hence, it is possible to use these indices to control the balance of exploration and exploitation in a PSO algorithm. In this section, some aspect of the usage of these indices is discussed.

The first aspect is the computational effort of using these indices. Including the indices in a PSO algorithm will increase computational effort if the indices are measured in every iteration. To reduce this effort, it could be measured only every n iterations, say 50 or 100 iterations.

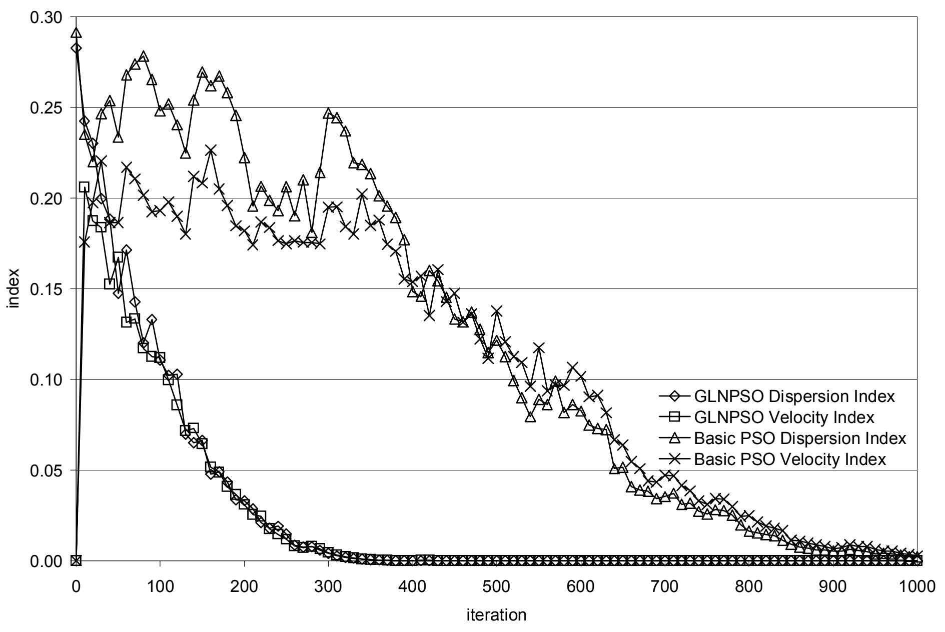

Fig. 5. Dispersion and Velocity Index on Typical Run of Basic PSO and GLNPSO on Rosenbrock Function.

The second aspect is the similarity of patterns between dispersion and velocity indices. The result in Section 3 shows that these indices have similar pattern. Figure 5 shows the pattern more clearly for one typical run of solving Rosenbrock function. Since the patterns are similar, it is sufficient to use only one index in a PSO Algorithm. The velocity index is preferable since it has simpler formula than the dispersion index.

The third aspect is how to implement this index in a PSO Algorithm to achieve a balance between exploration and exploitation processes. A simple idea is to ensure that the exploration and exploitation processes are performed in the same number of iterations. The exploration process is assumed finished when the velocity index reached a very small value, i.e. $\bar{\upsilon} \leq \varepsilon$. After number of iterations in the exploration process (τ) is known, the exploitation process is performed with exactly the same number of iterations (τ). Hence, the iteration process is stop when $t = 2\tau$. In this way, the exploration and exploitation processes are balanced by means of the same number of iterations.

Table 2 shows the solutions of the benchmark problems by GLNPSO with modified stopping criterion using two values of ε : 1E-03 and 1E-05. It shows that the smaller the value of ε, the higher total the number of iteration performed. However this increment tends to improve the solution quality. For these benchmark functions: this increment brings the objective function closer to zero.

The exploration and exploitation pattern of the swarm may depend on different test problems or different PSO algorithms or different parameter setting. Using the proposed stopping criterion with the small enough value of ε will allow enough iteration for exploitation process. Since it is already proven that the balance between exploration and exploitation may lead to a good solution, this stopping criterion may overcome the problem of finding the best parameter setting of algorithm for a certain problem. However, the effectiveness of a certain algorithm or parameter setting still could be comparable by the number of iteration needed to reach a certain level of velocity index.

V. Conclusion

The proposed dispersion and velocity indices could be used as a tool to monitor the balance of exploration and exploitation processes in PSO algorithm. After embedding these indices to a PSO algorithm, these indices could also be used to control the balance between exploration and exploitation processes in the algorithm, i.e. using velocity index to indicate the completeness of exploration process and perform the exploration and exploitation processes in the same number of iterations. A further study is required to observe behavior of other PSO algorithms and also other benchmark functions using the dispersion and velocity index for the generalization of this result. The PSO with the modified stopping criterion also need to be further explored. One aspect that important to be studied is the recommendation value of ε.

TABLE II
TYPICAL RESULT OF THE GLNPSO WITH MODIFIED STOP CRITERION
(A) OBJECTIVE FUNCTION VALUE

Function	Parabola		Griewank		Rosenbrock	
Replication	ε = 1E-03	ε = 1E-05	ε = 1E-03	ε = 1E-05	ε = 1E-03	ε = 1E-05
1	4.77E-17	3.89E-21	1.23E-02	7.39E-03	2.53E+01	2.31E+01
2	4.30E-16	9.81E-20	4.90E-02	7.39E-03	2.43E+01	2.43E+01
3	4.03E-15	1.70E-17	7.39E-03	1.23E-02	7.79E+01	2.29E+01
4	2.30E-16	7.61E-19	7.39E-03	7.39E-03	7.79E+01	2.09E+01
5	4.12E-16	4.57E-19	4.42E-02	4.93E-03	8.27E+01	1.88E+01

Function	Alpine		Ackley		Rastrigin	
Replication	ε = 1E-03	ε = 1E-05	ε = 1E-03	ε = 1E-05	ε = 1E-03	ε = 1E-05
1	1.42E-01	2.69E-03	1.16E+00	3.92E-12	4.38E+01	3.88E+01
2	2.42E-05	4.83E-04	1.34E+00	1.70E-08	8.26E+01	5.97E+01
3	1.88E-03	1.36E-03	9.31E-01	8.81E-09	4.28E+01	3.28E+01
4	4.64E-06	1.83E-04	7.07E-08	3.29E-09	5.07E+01	3.88E+01
5	3.30E-04	2.16E-05	4.91E-08	4.76E-08	5.37E+01	5.27E+01

(B) TOTAL NUMBER OF ITERATION

Function	Parabola		Griewank		Rosenbrock	
Replication	ε = 1E-03	ε = 1E-05	ε = 1E-03	ε = 1E-05	ε = 1E-03	ε = 1E-05
1	700	900	700	900	800	1400
2	700	900	700	1000	700	1400
3	700	900	700	1000	800	1400
4	700	900	700	900	800	1400
5	700	1000	700	900	800	1400

Function	Alpine		Ackley		Rastrigin	
Replication	ε = 1E-03	ε = 1E-05	ε = 1E-03	ε = 1E-05	ε = 1E-03	ε = 1E-05
1	1100	1100	700	1000	900	1000
2	800	1000	700	1000	1000	1100
3	800	1100	700	900	900	1200
4	800	1000	700	900	800	1100
5	800	1100	700	1000	900	1000

VI. ACKNOWLEDGEMENT

The research is part of the research program of the High Performance Computing Group at Asian Institute of Technology (AITHPC) which received support from the Thai GRID project. The authors wish to thank the AITHPC and the Thai GRID Center for the access of computing facility and the technical support.

REFERENCES

[1] J. Kennedy and R. Eberhart. "Particle swarm optimization," in *Proc. 1995 IEEE Int. Conf. Neural Networks*, vol. 4, pp. 1942–1948.

[2] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*. Morgan Kaufmann, 2001.

[3] M. Clerc, *Particle Swarm Optimization*. London: ISTE, 2006.

[4] Y. H. Shi and R. Eberhart, "Empirical study of particle swarm optimization," in *Proc. 1999 Congress on Evolutionary Computation*, pp. 1945-1950.

[5] X. H. Hu and R. Eberhart, "Solving constrained nonlinear optimization problems with particle swarm optimization," in *Proc. 6th World Multiconference on Systemics, Cybernetics and Informatics, 2002*.

[6] K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization method for constrained optimization problems," *Proc. 2002 Euro-International Symposium on Computational Intelligence*.

[7] E. C. Laskari, K. E. Parsopoulos and M. N. Vrahatis, "Particle swarm optimization for integer programming," *Proc. 2002 Congress on Evolutionary Computation*, pp. 1582–1587.

[8] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection," *Information Processing Letters*, vol. 85, no. 6, pp. 317–325, Mar. 2003.

[9] K. Veeramachaneni, T. Peram, C. Mohan and L. A. Osadciw, "Optimization using particle swarms with near neighbor interaction," *Proc. 2003 Genetic and Evolutionary Computation Conference*, pp. 110–121.

[10] F. Wang, N. Feng and Y. Qiu, "An adaptive diversity strategy for particle swarm optimization," *Proc. 2005 IEEE Int. Conf. Natural Language Processing and Knowledge Engineering*, pp. 760–764.

[11] P. Pongchairerks and V. Kachitvichyanukul, "A non-homogenous particle swarm optimization with multiple social structures," *Proc. 2005 Int. Conf. Simulation and Modeling*, paper A5-02.

[12] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," *Proc. 2000 Congress on Evolutionary Computation*, vol. 1, pp. 84–88.

[13] M. Clerc and J. Kennedy, "The particle swarm: explosion stability and convergence in a multi-dimensional complex space," *IEEE Transaction on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, Feb. 2002.