Adaptive Particle Swarm Optimizer: Response to Dynamic Systems through Rank-based Selection

Tauheed Ahmed, Farhad Kamangar

Department of Computer Science and Engineering
The University of Texas at Arlington
Arlington, TX 76019-0015
{tahmed, kamangar}@cse.uta.edu

Abstract

A response method to dynamic changes based on evolutionary computation is proposed for the particle swarm optimizer. The method uses rank-based selection to replace half of the lower fitness population with the higher fitness population, when changes are detected. Time-varying values for the acceleration coefficients are proposed to keep a higher degree of global search and a lower degree of local search at the beginning stages of the search. Performance is compared with two previous response methods using the parabolic De Jong benchmark function. Experimental results on the function with varying severity and dynamic change frequency is analyzed.

Introduction

The world around us is constantly changing. As a result, most real-world optimization problems are dynamic in nature. One of the recent approaches to many optimization problems in engineering and telecommunication has been the use of a method known as the Particle Swarm Optimization (PSO), which is based on the concept of Swarm Intelligence. Although, this method has been mostly applied to static optimization problems, some recent research has proved PSO to be effective in dynamic environments as well.

Particle Swarm Optimization (PSO) was originally developed and introduced by a social-psychologist named James Kennedy and an electrical engineer named Russell Eberhart [Kennedy and Eberhart 1995, Eberhart and Kennedy 1995]. PSO is a population based search optimization algorithm where each particle has a position, velocity, and a memory that keeps track of the previous position that evaluated to the best fitness. Particles in the swarm adjust their velocity based on their momentum and both individual and global memory. The stochastic nature of the algorithm keeps particles from falling into fixed routes. The original form of the particle swarm optimizer is defined by the following equation:

$$v_{id} \leftarrow v_{id} + \rho_1 r_1 (p_{id} - x_{id}) + \rho_2 r_2 (p_{gd} - x_{id}) \quad (1a)$$

$$x_{id} \leftarrow x_{id} + v_{id} \quad (1b)$$

where, v_{id} is the velocity of particle *i* along dimension *d*, x_{id} is the position of particle *i* in dimension *d*, ρ_1 and ρ_2 are acceleration coefficients, also known as the cognitive and social learning rates. p_{id} is the particle's previous best position (pbest) and p_{gd} is the best position found in the particle's neighborhood, r_1 and r_2 are two random functions in the range [0, 1] and *g* is the index of the best fitness particle in the neighborhood. The PSO implementation also includes the boundaries of the search space (Xmin, Xmax) and a limit on the particle velocity (Vmax). In [Kennedy 1997] four variations of the PSO model are described. The model with equal influences on both cognitive and social component is termed as the "full model". The model with no cognitive influence is called the "social-only model", the "cognition-only model" has no social influence and the "selfless model" is the same as the "social-only model" with the constraint that the calculation of a particle's neighborhood best does not include the particle itself. A new parameter for the PSO was introduced by Shi and Eberhart [Shi and Eberhart 1998] called the inertia weight, w, which was intended to dampen the velocity over time and thus allowing convergence with higher precision. The velocity formula with the inertia weight is as follows:

$$v_{id} \leftarrow w * v_{id} + \rho_1 r_1 (p_{id} - x_{id}) + \rho_2 r_2 (p_{gd} - x_{id}) \quad (2)$$

To reduce the undesirable explosive nature of the particles, Clerc [Clerc 1999] derived a constriction coefficient K, which eliminated the need for a Vmax. The constriction factor is computed as follows:

$$K = \frac{2}{|\, 2 - \rho - \sqrt{\rho^2 - 4\rho}\,|} \quad (3)$$

where, $\rho = \rho_1 + \rho_2$, $\rho > 4$ and the velocity computations are done using the following equation:

$$v_{id} \leftarrow K(v_{id} + \rho_1 r_1 (p_{id} - x_{id}) + \rho_2 r_2 (p_{gd} - x_{id})) \quad (4)$$

Recently a "fully informed" [Mendes, Kennedy and Neves 2004] PSO was proposed by making all neighboring particles influence the velocity calculation, instead of just using the influence of the best fitness particle.

Various methods have been applied to the Standard Particle Swarm Optimizer to adapt the algorithm for dynamic environments. Carlisle and Dozier used the change in one or more randomly picked particle's fitness value as a sign of change in the system and replaced the previous best vector (P) by the current position vector (X) to adapt PSO to the dynamic changes [Carlisle and Dozier 2002]. The change in the global optimum location or the absence of change in the global optimum location for some consecutive iterations was used by Hu and Eberhart as a sign of change in the system and various re-randomization methods were used to respond to the changes [Hu and Eberhart 2002]. Parrott and Li used speciation and crowding [Parrott and Li 2004] to create sub-population in parallel to track multiple peaks in a dynamic environment using PSO.

In this paper, an adaptive particle swarm optimizer is proposed that uses rank-based selection to respond to dynamic changes. Variable values for the acceleration coefficients are also proposed for improved search.

Dynamic Systems

The state of a dynamic system can change continuously, at fixed intervals or just randomly. Eberhart and Shi defined three kinds of dynamic systems [Eberhart and Shi 2001] based on the way they change over time. They are: a system where the location of the optimum changes, a system where the location of the optima remains constant but the value of the optima changes, and a system where both the optimum location and the value changes. Also in a multidimensional system, the change can happen in one or more dimensions [Hu and Eberhart 2002]. Dynamic systems can also be categorized according to the type of changes. Changes can be classified based on dynamic frequency, severity or predictability of the change [Branke 1999].

In this paper, investigation is done on systems where the location of the optimum changes. Dynamic environments are simulated by applying various temporal dynamics to the objective function using two variables called 'severity' and 'dynamic change frequency'. Severity refers to the amount of displacement of the objective function from its current position. With increasing severities, the swarm has less time to catch-up to the optima as it gets less time to learn and propagate the updates. Dynamic change frequency indicates the number of generations between each displacement. Higher values of this variable allow the swarm to have more time to react to changes and thus result in a higher success rate.

PSO with Rank-based Selection (RS-PSO)

Selection [Bäck, Fogel and Michalewicz 2000] is the process of choosing individuals for reproduction or survival in an evolutionary algorithm. The main objective of the selection operator is to emphasize better solutions in a population. Selection doesn't create any new solution, but selects relatively good solutions from a population and deletes the remaining solutions. A particular solution's fitness is used to identify how good or bad the solution is compared to the neighboring solutions in the search space. The idea is to assign higher probability of selection to solutions having better fitness values. There are various ways of implementing selection: proportional or roulette wheel, boltzmann, tournament, rank-based, soft brood, disruptive, competitive selection, etc. In this paper, linear rank-based selection was chosen as the response method for dynamic changes for the particle swarm optimizer.

Linear ranking selection assigns a selection probability to each individual that is proportional to the individual's rank. In linear ranking, the selection probability for individual *i* is defined as follows [Bäck, Fogel and Michalewicz 2000]:

$$p_{lin\_rank}(i) = \frac{\alpha_{rank} + \frac{rank(i)}{\mu - 1} * (\beta_{rank} - \alpha_{rank})}{\mu} \quad (5)$$

where, α_{rank} is the number of offspring allocated to the worst individual, β_{rank} is the expected number of offspring to be allocated to the best individual and μ is the population size. Some of the major advantages of linear rank selection over other selection methods are:

(1) Rank-based selection is simple and easy to implement,
(2) This selection gives more selective pressure towards the optimum when the fitness values of the individuals are similar,
(3) One of the major drawbacks of proportional selection is the presence of a 'super' individual, which might completely take over the population in a single generation because of its vastly superior fitness value. Rank-based selection can avoid premature convergence caused by 'super' individuals, since the best individual is always assigned the same selection probability, regardless of its fitness value.

Along with rank-based selection, this paper implemented variable values for the acceleration coefficients in different stages of the search. The idea was mostly influenced by the concept of time decreasing values for the inertia weight [Shi and Eberhart 1998]. The acceleration coefficients in the velocity formula control the amount of global or local search. Kennedy recommended [Kennedy, Eberhart 1995] the value of 2 for the

coefficients, as it makes the weights for social and cognition parts to be 1 on the average. The rationale behind the use of variable coefficients was to make the particles move around in a larger area in the early stages of the search process, by giving higher weight to the cognition component. In the same way, a higher weight for the social component in the latter stages of the search was used to make the particles shrink their search in a smaller area for a fine grained search. To give ρ_1 (cognitive learning rate) a higher initial value and ρ_2 (social learning rate) a lower initial value, the following equations were used:

$$\rho_1 = \rho_1 init + \frac{current\_iteration}{Max\_allowed\_iteration} * (\rho_1 final - \rho_1 init) \quad (6a)$$

$$\rho_2 = \rho_2 init + \frac{current\_iteration}{Max\_allowed\_iteration} * (\rho_2 final - \rho_2 init) \quad (6b)$$

where, ρ_{1init} and ρ_{1final} are initial and final values of the cognitive learning rate, ρ_{2init} and ρ_{2final} are initial and final values of the social learning rate. Similar approach has been used recently [Ratnaweera, Halgamuge and Watson 2004] to find faster convergence in static optimization problems. Their experimental results showed that, the initial value of 2.5 and the final value of 0.5 for ρ_1 and the initial value of 0.5 and the final value of 2.5 for ρ_2 is a good choice for most benchmark functions. The same values were adopted for the experiments in this paper.

Experiments

The proposed model (RS-PSO) was tested in various severities and dynamic change frequencies. For the purpose of comparative analysis, two previously [Carlisle and Dozier 2002, Eberhart and Shi 2001] proposed PSO models were used, which are explained later in the implementation section.

Design

Whenever changes were detected in the system, the particles were given a rank, based on their fitness value, with the lowest fitness particle getting a rank value of 1. Then the entire population was sorted according to the rank value and the position of the best half of the population with higher ranks were used to replace the position of the lower half of the population with lower ranks. Angeline used a similar method [Angeline 1998] to improve the standard particle swarm optimizer in static environment. But he used tournament selection instead of the rank-based selection where, a randomly chosen portion of the population, instead of the entire population (as in rank-based selection), was considered at each step. Also, as he was investigating static optimization problems, there were no dynamic changes involved and replacements were done at every iteration.

The rank-selection based replacement of the worse half of the population ensures that in case of changes in the system, half of the population would be moved to positions of the search space that resulted in better fitness values. The replacement wouldn't destroy the memory of the particles as the values of personal best were unchanged. The objective of this method is to arm the optimizer with a more exploitive search.

The parabolic De Jong function, Equation (7), was chosen as the benchmark (objective) function. This selection, was made based on Angeline's experiment [Angeline 1997].

$$f(x) = \sum_{i=1}^{3} (x_i - offset)^2 \quad (7)$$

The movement of the optimum at various intervals was controlled using the dynamic parameter *offset*. Only linear motion was tested in the experiments. The position of the particles was updated at specified intervals as $x_d = x_d + s$, where s is the severity and x_d is the current position in dimension d. This added a constant displacement to the position of the particles in each dimension equivalent to the severity.

Implementation

A Matlab program was developed to experiment with PSO in various parameter settings. The parameters of the two previously proposed PSO models [Carlisle and Dozier 2002, Eberhart and Shi 2001] were kept as close to their original setting as possible for a fair comparison.

Model-1: This model was designed according to Carlisle and Dozier's experiment [Carlisle and Dozier 2002]. The search space was restricted to [-50, 50] on all dimensions, population size was 30, initially randomly distributed over [-50, 50] in each dimension. The inertia weight was a linearly decreasing function, initially set to 0.65, decreasing to 0.15 over the first 2/3 of the maximum allowed iteration and remained at 0.15 later. The acceleration coefficients were set to 2.05, Clerc's [Clerc 1999] constriction factor was used, no Vmax was used and the maximum allowed iteration was 500. Maximum allowed error was set to 0.001. Average of 30 runs for each experiment was used. In this model, *sentry* particles were used to detect changes in the system and each particle's P vector was replaced with the X vector to respond to the dynamic changes.

Model-2: The parameters of this model were set closely to Hu and Eberhart's [Hu and Eberhart 2002] model. The search space was restricted to [-50, 50] on all dimensions, the population size was 30, initially randomly distributed over [-50, 50]. The inertia weight was 0.5 + Rnd/2.0. The maximum velocity was set to the dynamic range of each dimension. Maximum allowed error was set to 0.001. Acceleration coefficients were set to 2. "changed-gbest-value" method [Hu and Eberhart 2002] was used to detect changes in the system. Among the various re-randomization methods applied by Hu and Eberhart, re-

randomization of 10% of the population was reported to be a good choice. The same was used for the purpose of this research.

When comparing the proposed PSO model with the other two models, only the response method (rank-based selection) and acceleration coefficients (variable) differed, while the other parameters were kept the same as the model compared with.

While presenting the experimental results, the model proposed in this paper would be referred to as RS-PSO-1 when using the parameters of Model-1 and RS-PSO-2 when using parameters of Model-2. The tables and figures used the label '*f*' to refer to the dynamic change frequency in the results.

Results

All experiments were repeated for the severity values of 0.00001, 0.00005, 0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5 and 1.0. The values for the dynamic change frequency were 1, 5, 10, 15, 20, 25, 30, 35 and 40. Result of 30 runs for each parameter setup was recorded. The reliability of the algorithm was recorded in terms of solutions found (% solved) and the efficiency was recorded in terms of average iterations per solution found. The results showed that the proposed PSO model successfully responded to dynamic changes and was able to track the optimum. The results also showed improvements over both the previous models in most cases. Only selected values (randomly picked) for severity and dynamic change frequency are presented in this paper.

Table 1: Percentage of success (reliability) of RS-PSO-1 in various change frequency

Severity	% solved				
	f=1	f=10	f=20	f=30	f=40
0.00001	100	100	100	100	100
0.00005	100	100	100	100	100
0.0001	100	100	100	100	100
0.0005	100	100	100	100	100
0.001	100	100	100	100	100
0.005	70	100	100	100	100
0.01	40	100	100	100	100
0.05	0	75	95	100	100
0.1	0	20	80	90	90
0.5	0	0	15	40	35
1.0	0	0	5	5	40

Table-1 shows the percentage of success of the RS-PSO-1 model (the proposed methods with Model-1 parameter settings) in various severities and various change frequency. Here, percentage of success (reliability) is defined by the percentage of success out of 30 runs, for each pair of severity and change frequency. The values show that with increasing values of change frequency, the performance of RS-PSO-1 increased, as with higher frequency, it was able to find solutions with higher severity (as it gets more time to adapt). Figure-1 represents the Table-1 data in graph format.

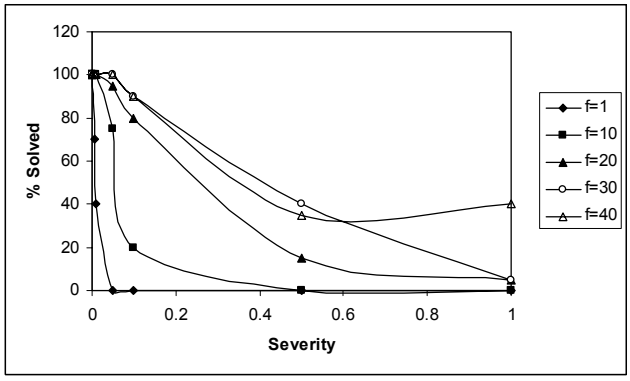

Figure 1: Comparative performance of RS-PSO-1 in various dynamic change frequency

Table 2: Percentage of success (reliability) of RS-PSO-2 in various change frequency

Severity	% solved				
	f=1	f=10	f=20	f=30	f=40
0.00001	100	100	100	100	100
0.00005	100	100	100	100	100
0.0001	100	100	100	100	100
0.0005	100	100	100	100	100
0.001	100	100	100	100	100
0.005	95	100	100	100	100
0.01	30	100	100	100	100
0.05	0	75	100	100	100
0.1	0	10	65	95	100
0.5	0	0	25	15	30
1.0	0	0	0	0	15

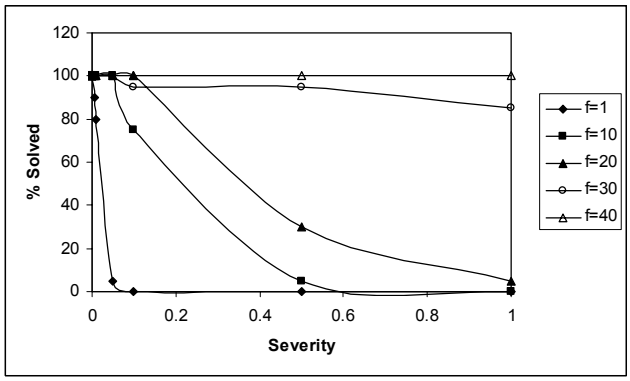

Figure 2: Comparative performance of RS-PSO-2 in various dynamic change frequency

Table-2 shows the percentage of success of the model RS-PSO-2 (the proposed methods with Model-2 parameter settings) in various severities and various change frequency. Like RS-PSO-1 model, this model also improved it's percentage of success with higher change frequency. Figure-2 shows the same result in a graph.

Figure-3 shows the comparison between Model-1 and RS-PSO-1 with respect to average number of iteration required to find the optimum in various severities in

change frequencies of 5 and 20. For change frequency 5, RS-PSO-1 requires lesser iterations to find the optimum than Model-1 when severity is smaller. At frequency 20, the performance difference is more visible as RS-PSO-1 finds the optima within only 50 iterations.

Figure 4 shows the comparative performance of Model-1 and RS-PSO-1 with respect to percentage of success of finding the optimum in various change frequencies. RS-PSO-1 shows a higher percentage of success in all three change frequencies.

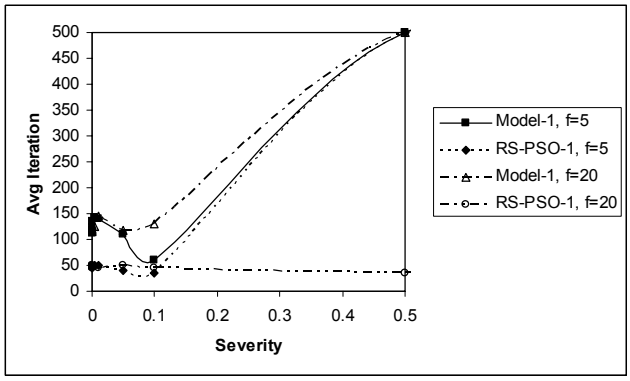

Figure 3: Avg Iteration vs. Severity at dynamic change frequency = 5 and 20

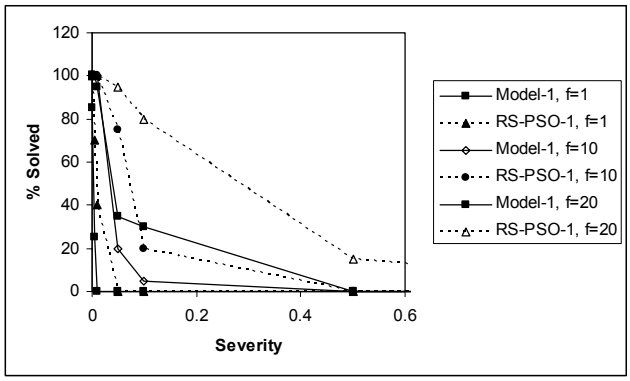

Figure 4: Comparison of percentage of success at dynamic change frequency = 1, 10 and 20

Figure-5 shows the comparison between Model-2 and RS-PSO-2 with respect to average number of iteration required to find the optimum in various severities, in change frequencies of 1 and 15. For both frequency, RS-PSO-2 out performs Model-2.

Figure 6 shows the comparative performance of Model-2 and RS-PSO-2 with respect to percentage of success of finding the optimum in various change frequencies. RS-PSO-2 shows higher degree of success in finding the optimum for higher severities than Model-2.

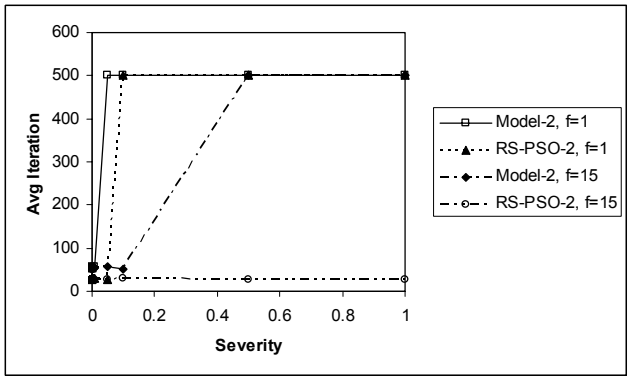

Figure 5: Avg Iteration vs. Severity at change frequency = 1 and 15

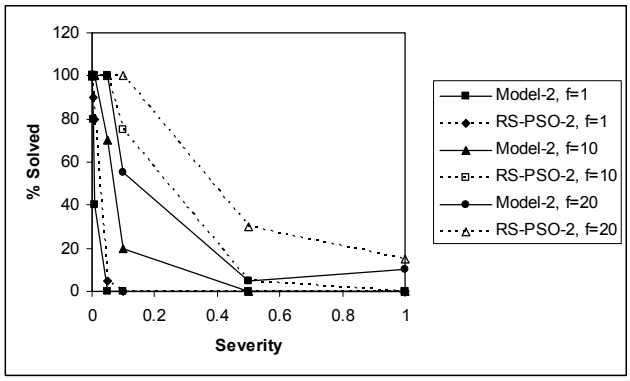

Figure 6: Comparison of percentage of success at change frequency = 1, 10 and 20

Conclusion

A new response method to dynamic changes was proposed for the particle swarm optimizer, where half of the swarm with lower fitness was replaced with the higher fitness particles. Constant values of the acceleration coefficients were replaced with variable values to implement a more effective search that takes into consideration, the amount of local or global search needed in different stages of the search.

Performance of the proposed PSO model (RS-PSO) was compared with two other previously proposed PSO models using the parabolic De Jong function. Experimental results under various severities and change frequencies showed that the proposed model was successful at tracking the optima of a dynamically changing function. The combination of variable acceleration coefficients and response through rank-based selection showed better performance in both the number of iteration required to converge (efficiency) and in the probability of success, compared to the previous two methods. Further experiments are required to determine to what degree the

two factors (rank selection and variable acceleration coefficient) individually contribute to the performance gain. The authors of this paper acknowledge the fact that various other factors may have contributed to the performance of the various PSO models tested and thus do not claim to have proposed a superior PSO model than the other models. Instead, this paper presents the preliminary results and proposes rank-based selection and variable acceleration coefficients as effective methods for dynamic optimization problems.

Future Research

A simplistic dynamic system was simulated in this research that applied uniform linear motion in a simple continuous unimodal function. For applicability in a wide range of real-world optimization problems, the proposed methods need to be tested with highly chaotic motion in multimodal functions. As an enhancement to the variable acceleration coefficients method, a relation can be devised between the coefficients and the overall performance of the swarm, so that, the weight of the cognitive and the social component can be proportionally increased or decreased depending on the need, which could be calculated at any point in the search from the fitness values.

References

Angeline, P. J. 1997. Tracking extrema in dynamic environments. In *Proceedings of the 6th International Conference on Evolutionary Programming*, Apr. 13-16. Indianapolis, Indiana, USA, P. J. Angeline, R. G. Reynolds, J. R. McDonnell, and R. C. Eberhart, Eds., vol. 1213 of Lecture Notes in Computer Science, Springer Verlag.

Angeline, P. J. 1998. Using selection to improve particle swarm optimization. In *IEEE International Conference on Evolutionary Computation*, IEEE Press.

Bäck, T., and Fogel, D. B., and Michalewicz, T. Eds. 2000. Evolutionary Computation 1: Basic Algorithms and Operators, Institute of Physics Publishing, Bristol and Philadelphia.

Branke, J. 1999. Evolutionary algorithms for dynamic optimization problems: A survey. Tech. Rep. 387, Institute AIFB, University of Karlsruhe, Karlsruhe, Germany, 1999.

Carlisle, A., and Dozier, G. 2000. Adapting particle swarm optimization to dynamic environments. In *International Conference on Artificial Intelligence*, Monte Carlo Motel, Las Vegas, NV, USA, 2000, vol. 1, pp. 429-434.

Carlisle, A., and Dozier, G. 2001. Tracking changing extrema with particle swarm optimizer. Tech. Rep. CSSE01-08, Auburn University, Auburn, AL, USA, 2001.

Carlisle, A., and Dozier, G. 2002. Tracking changing extrema with adaptive particle swarm optimizer. In ISSCI,024 (*2002 World Automation Congress*, Orlando, FL, USA, 2002), vol. 3.

Clerc, M. 1999. The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. In *Proceedings of the Congress on Evolutionary Computation* (Mayflower Hotel, Washington D.C., USA, 6-9 July 1999), P. J. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala, Eds., vol. 3, IEEE Press, pp. 1951-1957.

Eberhart, R. C., and Kennedy, J. 1995. A new optimizer using particle swarm theory. In *Sixth International Symposium on Micro Machine and Human Science* (Nagoya, Japan, 1995), IEEE Press, pp. 39-43.

Eberhart, R. C., and Shi, Y. 2001. Tracking and Optimizing Dynamic Systems with Particle Swarms. In the *Proceedings of the Congress on Evolutionary Computation 2001*, Piscataway, NJ: IEEE Press, pp. 94-97, 2001.

Hu, X., and Eberhart, R. C. 2002. Adaptive Particle Swarm Optimization: Detection and Response to Dynamic Systems. *Proceedings of the Congress on Evolutionary Computation*, 2002, pp. 1666-1670. Hawaii.

Kennedy, J. 1997. The particle swarm: Social adaptation of knowledge. *IEEE International Conference on Evolutionary Computation* (Indianapolis, Indiana), IEEE Service Center, Piscataway, NJ, 303-308.

Kennedy, J., and Eberhart, R. C. 1995. Particle swarm optimization. *IEEE International Conference on Neural Networks* (Perth, Australia), IEEE Service Center, Piscataway, NJ, IV: 1942-1948.

Mendes, R., and Kennedy, J., and Neves, J. 2004. The Fully Informed Particle Swarm: Simpler, Maybe Better. *IEEE Transactions on Evolutionary Computation*, 8(3), pp. 204-210, 2004.

Parrott, D., and Li, X. 2004. A Particle Swarm Model for Tracking Multiple Peaks in a Dynamic Environment using Speciation. *Proceedings of the 2004 Congress on Evolutionary Computation* (CEC'04), pp. 98-103, IEEE Service Center, Piscataway, NJ 08855-1331.

Ratnaweera, A., and Halgamuge, S. K., and Watson, H. C. 2004. Self-Organizing Hierarchical Particle Swarm Optimizer With Time-Varying Acceleration Coefficients. *IEEE Transaction on Evolutionary Computation*, 8(3), pp. 240-253, 2004.

Shi, Y., and Eberhart, R. C. 1998. Parameter selection in particle swarm optimization. In *Proceedings of the 7th Annual Congress on Evolutionary Programming*, San Diego, USA.

Shi, Y. H., and Eberhart, R. C. 1998 A modified particle swarm optimizer. In *IEEE International Congress on Evolutionary Computation* Piscataway, NJ: IEEE Press, pp. 69-73, 1998.