A Review on Filter Design Strategies Using Optimization Techniques

Vaibhav Srivastava, Ramapati Mishra
Department of Electronics & Communication
IET, Dr. RMLAU, Ayodhya (U.P.)

Abstract: The particle swarm paradigm, that was only a few years ago a curiosity, has now attracted the interest of researchers around the globe. This article is intended to give an overview of important work that gave direction and impetus to research in particle swarms as well as some interesting new directions and applications. Things change fast in this field as investigators discover new ways to do things, and new things to do with particle swarms. It is impossible to cover all aspects of this area within the strict page limits of this journal article. Thus this paper should be seen work of few authors have at the time of writing.

Keywords-Pass band, Stop band ripples, PSO, Velocity formula.

1. Introduction:

Introduced by Eberheart and Kennedy in 1995 [2], PSO is a search technique based on social behavior of bird flocking and fish schooling. There are different kinds of bio and social behavior inspired algorithms. PSO is one of the different swarm based algorithms. In PSO, each particle of the swarm is a possible solution in the multi-dimensional search space. The particles change their positions with a certain velocity in each iteration, according to the standard PSO equations, thus moving towards the global best (gbest) solution. Being easy to implement and yet so effective, PSO has been utilized in a wide variety of optimization applications. Particle swarm optimization is a population based search algorithm and is inspired by the observation of natural habits of bird flocking and fish schooling. In PSO, a swarm of particles moves through a D dimensional search space. The particles in the search process are the potential solutions, which move around the defined search space with some velocity until the error is minimized or the solution is reached, as decided by the fitness function. Fitness function is the measure of particles fitness which is the deviation of the particle from the required solution. The particles reach to the desired solution by updating their position and velocity according to the PSO equations. In PSO model, each individual is treated as a volume-less particle in the D-dimensional search space with initial random velocity.

A filter is a frequency selective circuit that allows a certain frequency to pass while attenuating the others. Filters could be analog or digital. Analog filters use electronic components such as resistor, capacitor, transistor etc. to perform the filtering operations. These are mostly used in communication for noise reduction, video/audio signal enhancement etc. In contrast, digital filters use digital processors which perform mathematical calculations on the sampled values of the signal in order to perform the filter operation. A computer or a dedicated digital signal processor may be used for implementing digital filters. Filters mostly find their use in communication for noise reduction, audio/video signal enhancement etc.

2. Related Work:

The initial ideas on particle swarms of Kennedy (a social psychologist) and Eberhart (an electrical engineer) were essentially aimed at producing computational intelligence by exploiting simple analogues of social interaction, rather than purely individual cognitive abilities. The first simulations (Kennedy and Eberhart 1995) were influenced by Heppner and Grenander's work (Heppner and Grenander 1990) and involved analogues of bird flocks searching for corn. These soon developed (Kennedy and Eberhart 1995; Eberhart and Kennedy 1995; Eberhart et al. 1996) into a powerful optimization method— Particle Swarm Optimization (PSO).

In PSO a number of simple entities—the particles—are placed in the search space of some problem or function, and each evaluates the objective function at its current location. Each particle then determines its movement through the search space by combining some aspect of the history of its own current and best (best-fitness) locations with those of one or more members of the swarm, with some random perturbations. The next iteration takes place after all particles have been moved. Eventually the swarm as a whole, like a flock of birds collectively foraging for food, is likely to move close to an optimum of the fitness function.

The evolutionary computation (EC) community has shown a significant interest in optimization for many years. In particular, there has been a focus on global optimization of numerical, real-valued 'black-box' problems for which exact and analytical methods do not apply. Since the mid-sixties many general-purpose optimization algorithms have been proposed for finding near-optimal solutions to this class of problems; most notably: evolution strategies (ES) [8], evolutionary programming (EP) [3], and genetic algorithms (GA) [6].

Many efforts have also been devoted to compare these algorithms to each other. Typically, such comparisons have been based on artificial numerical benchmark problems. The goal of many studies was to verify that one algorithm outperformed another on a given set of problems. In general, it has been possible to improve a given standard method within a restricted set of benchmark problems by making minor modifications to it.

Recently, particle swarm optimization (PSO) [7] and differential evolution (DE) [11] have been introduced and particularly PSO has received increased interest from the EC community. Both techniques have shown great promise in several real-world applications [4], [5], [12], [14]. However, to our knowledge, a comparative study of DE, PSO, and Gas on a large and diverse set of problems has never been made. In this study, we investigated the performance of DE, PSO, and an evolutionary algorithm (EA) 1 on a selection of 34 numerical benchmark problems. The main objective was to

examine whether one of the tested algorithms would outperform all others on a majority of the problems. Additionally, since we used a rather large number of benchmark problems, the experiments would also reveal whether the algorithms would have any particular difficulties or preferences.
Overall, the experimental results show that DE was far more efficient and robust (with respect to reproducing the results in several runs) compared to PSO and the EA. This suggests that more emphasis should be put on DE when solving numerical problems with real-valued parameters. However, on two noisy test problems, DE was outperformed by the other algorithms.
Kennedy and Spears (1998) compared this binary particle swarm to several kinds of GAs, using Spears' multimodal random problem generator. This paradigm allows the creation of random binary problems with some specified characteristics, e.g., number of local optima, dimension, etc. In that study, the binary particle swarm was the only algorithm that found the global optimum on every single trial, regardless of problem features. It also progressed faster than GAs with crossover, mutation, or both, on all problems except the very simplest ones, with low dimension and a small number of local optima; the mutation-only GA was slightly faster in those cases.
Several other researchers have proposed alterations to the particle swarm algorithm to allow it to operate on binary spaces. Agrafiotis and Cedeño [1] used the locations of the particles as probabilities to select features in a pattern-matching task. Each feature was assigned a slice of a roulette wheel based on its floating-point value, which was then discretized to {0, 1}, indicating whether the feature was selected or not. Mohan and Al-Kazemi (2001) suggested several ways that the particle swarm could be implemented on binary spaces. One version, which he calls the "regulated discrete particle swarm," performed very well on a suite of test problems. In Pamparä et al. (2005), instead of directly encoding bit strings in the particles, each particle stored the small number of coefficients of a trigonometric model (angle modulation) which was then run to generate bit strings.

Extending PSO to more complex combinatorial search spaces is also of great interest. The difficulty there is that notions of velocity and direction have no natural extensions for TSP tours, permutations, schedules, etc. Nonetheless, progress has recently been made (Clerc 2004, 2006b; Moraglio et al. 2007) but it is too early to say if PSO can be competitive in these spaces.

Dynamic problems are challenging for PSO. These are typically modeled by fitness functions which change over time, rendering particle memory obsolete (Hu and Eberhart 2001).
Parsopoulos and Vrahatis (2001) showed that the particle swarm could track slowly moving optima without any changes at all. Eberhart and Shi (2001) made a slight adjustment to the inertia by randomizing the inertia weight between 0.5 and 1.0. The idea is that when tracking a single dynamic optimum, it can not be predicted whether exploration (a larger inertia weight) or exploitation (a smaller inertia weight) will be better at any given time.

However, in many cases more specialized changes are required to handle dynamic problems. The problem can be split into two: how to detect change and how to respond. Carlisle and Dozier, Carlisle and Dozier (2000, 2001) occasionally re-evaluate the previous best of a single particle. Change is indicated by a different function value upon re-evaluation. They tried two responses to change. In the first, the current position was set to be the previous best and in the second, and more successful response, previous bests are compared with current positions, and memory is updated accordingly. Hu and Eberhart (2002) used a similar scheme for change detection but, as a response, randomized the entire swarm in the search space.

As an alternative to these strategies, a level of diversity can be maintained throughout the run. Parsopoulos and Vrahatis (2004) use repulsion to keep particles away from detected optima. The authors of (Blackwell and Bentley 2002) introduced charged particles into the swarm. These particles mutually repel and orbit a converging nucleus of 'neutral' particles, in analogy with a (picture of) an atom. Charged swarms can detect optimum shifts within their orbit and are therefore able to track quite severe changes. Diversity can also be maintained with a grid-like local neighborhood (Li and Dam 2003) and using hierarchical structures (Janson and Middendorf 2004).
Dynamic multi-modal landscapes are especially challenging for PSO (Blackwell and Branke 2006). The strategies mentioned above—restart and diversity enhancement—are less successful in situations where function peaks can vary in height as well as location because a small change in peak height might entail a large change in the position of the global optimum. In these cases, multi-population approaches have proven to be beneficial. The idea behind multi-swarm models is to position swarms on different peaks so that, should a suboptimal peak become optimal, a swarm will be ready to immediately begin re-optimizing. Parrot and Li (2006) adjust the size and number of swarms dynamically by ordering the particles into 'species' in a technique called clearing. Blackwell and Branke (2006), borrowing from the atomic analogy referred to above, invoke an exclusion principle to prevent swarms from competing on the same peak and an anti-convergence measure to maintain diversity of the multi-swarm as a whole. Recently, a self-adapting multi-swarm has been derived (Blackwell 2007). The multi-swarm with exclusion has been favorably compared, on the moving peaks problem, to the hierarchical swarm, PSO re-initialization and a state of the art dynamic-optimization evolutionary algorithm known as self-organizing scouts.

Noisy fitness functions are important since they are often encountered in real-world problems. In these problems, unlike dynamic problems where the fitness function changes over time, the fitness function remains the same. However, its evaluation is noisy. Therefore, if a PSO explores the same position more than once, the fitness values associated to each evaluation may differ.
Parsopoulos and Vrahatis (2001) studied the behavior of the PSO when Gaussian distributed random noise was added to the fitness function and random rotations of the search space were performed. Experimental results indicated that the PSO remained effective in the presence of noise, and, in some cases, noise even helped the PSO avoid being trapped in local optima.
Pugh et al. (2005) compared the PSO to a noise-resistant variant where the main PSO loop was modified so that multiple evaluations of the same candidate solution are

aggregated to better assess the actual fitness of this particular solution. The comparison considered several numerical problems with added noise, as well as unsupervised learning of obstacle avoidance using one or more robots. The noise-resistant PSO showed considerably better performance than the original.

Several investigators have attempted to adapt PSO parameters in response to information from the environment. Techniques from evolutionary computation and other methods have been borrowed by particle swarm researchers as well.

Angeline [2] produced one of the first intentionally hybridized particle swarms. In his model, selection was applied to the particle population; "good" particles were reproduced with mutation, and "bad" particles were eliminated. Angeline's results showed that PSO could benefit from this modification.

Miranda and Fonseca (2002) borrowed an idea from evolution strategies. In that paradigm, points are perturbed by the addition of random values distributed around a mean of zero; the variance of the distribution is evolved along with function parameters. Those researchers used Gaussian random values to perturb χ, φ1, and φ2, as well as the position of the neighborhood best—but not the individual best—using selection to adapt the variance. The evolutionary self-adapting particle swarm optimization method, a hybrid of PSO and evolutionary methods, has shown excellent performance in comparison to some standard particle swarm methods. Miranda has used it for the manufacture of optical filters as well as in the optimization of power systems (Miranda and Fonseca 2002).

Loøvbjerg et al. (2001) use "breeding", borrowed from genetic algorithms, in a recent particle swarm study. Some selected particles were paired at random, and both positions and velocities were calculated from weighted arithmetic averages of the selected particles' parameters. Those researchers also divided the particle swarm into subpopulations in order to increase diversity, with some probability that individuals would breed within their own subpopulation or with a member of another. Results were encouraging, though the model as reported was not clearly superior to standard PSO or GA.

Wei et al. (2002) took a different tack, embedding velocity information in an evolutionary algorithm. They replaced Cauchy mutation with a version of PSO velocity in a fast evolutionary programming (FEP) algorithm, to give the FEP population direction. Their published results indicate that the approach is very successful on a range of functions; the new algorithm found global optima in tens of iterations, compared to thousands for the FEP versions tested.

Robinson et al. (2002), trying to optimize a profiled corrugated horn antenna, noted that a GA improved faster early in the run, and PSO improved later. As a consequence of this observation, they hybridized the two algorithms by switching from one to the other after several hundred iterations. They found the best horn by going from PSO to GA (PSO-GA) and noted that the particle swarm by itself outperformed both the GA by itself and the GA-PSO hybrid, though the PSO-GA hybrid performed best of all. It appears from their result that PSO more effectively explores the search space for the best region, while GA is effective at finding the best point once the population has converged on a single region; this is consistent with other findings.

Krink and Loøvbjerg (2002) similarly alternated among several methods, but they allowed individuals in a population to choose whether to belong to a population of a genetic algorithm, a particle swarm, or to become solitary hill-climbers. In their self-adaptive search method, an individual changed its stage after 50 iterations with no improvement. The population was initialized as PSO particles; the "LifeCycle" algorithm outperformed all three of the methods that comprised it. Krink and Loøvbjerg's graphs show interesting changes in the proportion of individuals in each state for various problems.

A hybrid between a PSO and a hill-climber was proposed by Poli and Stephens (2004) who considered swarms of particles sliding on a fitness landscape rather than flying over it. The method uses particles without memory and requires no book-keeping of personal best. Instead it uses the physics of masses and forces to guide the exploration of fitness landscapes. Forces include: gravity, springs, and friction. Gravity provides the ability to seek minima. Springs provide exploration. Friction slows down the search and focuses it.

Clerc's recent experiments (Clerc 2006b) have shown that adaptation of the constriction factor, population size, and number of neighbors can produce improved results. His studies found that best performance were obtained when all three of these factors are adapted during the course of the run. Clerc used three rules: (a) Suicide and generation: a particle kills itself when it is the worst in its neighborhood and generates a new copy of itself when it is the best; (b)Modifying the coefficient: good local improvement caused an increase in the constriction coefficient, while poor improvement caused its decrease; (c) Change in neighborhood: the locally best particle could reduce the number of its neighbors, while poorly performing particles could increase theirs. Adaptive changes were not made on every iteration, but only occasionally.

Vesterstrøm et al. (2002) borrowed the idea of division of labor from research on insect swarm algorithms. In their hybrid particle swarm model, individuals in the swarm were assigned, after some number of iterations without improvement, to conduct local search. Local search was implemented by placing a particle at the population's global best position with a new random velocity vector. The division of labor modification was intended to improve performance on unimodal problems; this improvement was seen, though performance on multimodal functions was not significantly improved.

Hendtlass (2001) proposed a hybridization of PSO with ant colony optimization (ACO) (Dorigo and Stützle 2004) but did not present any results. More recently, Holden and Freitas (2005) introduced a hybrid PSO/ACO algorithm for hierarchical classification. This was applied to the functional classification of enzymes, with very promising results.

Hendtlass (2001) combined PSO with differential evolution (DE) but with mixed results. While the hybridized algorithm did better than either PSO or DE on one multimodal problem, the particle swarm by itself tended to be faster and more robust than either DE or two version of hybrids that were tested. However, more recently, others, for example, Zhang and Xie (2003), have obtained more positive results.

A hybridization of PSO based on genetic programming (GP) was proposed (Poli et al. 2005b, 2005a) showing some promise. GP is used to evolve new laws for the control of particles' movement for specific classes of problems. The method has consistently provided PSOs that performed better than some standard reference PSOs in the problem class used for training, and, in some cases, also generalized outside that class.

A PSO that borrows from estimation of distribution algorithms has recently been proposed in (Iqbal and Montes de Oca 2006). In this approach, the swarm's collective memory is used to bias the particles' movement towards regions in the search space which are estimated to be promising and away from previously sampled low-quality regions. Experiments suggest that this PSO hybrid finds better solutions than the canonical PSO using fewer function evaluations.

Some researchers have noted a tendency for the swarm to converge prematurely on local optima. Several approaches have been implemented in order to correct for the decline of diversity as the swarm concentrates on a single optimum.

Loøvbjerg (2002) used self-organized criticality to help the PSO attain more diversity, making it less vulnerable to local optima. When two particles are too close to one another, a variable called the "critical value" is incremented. When it reaches the criticality threshold, the particle disperses its criticality to other particles that are near it and relocates itself.
Other researchers have attempted to diversify the particle swarm by preventing the particles' clustering too tightly in one region of the search space. Blackwell and Bentley (2002) collision-avoiding swarms achieve this by reducing the attraction of the swarm center.

Krink et al. (2002) developed "spatially extended" particles, where each particle is conceptualized as being surrounded by a sphere of some radius. When the spatially extended particle collides with another, it bounces off.

Xie et al. (2002) added negative entropy to the particle swarm in order to discourage premature convergence (excessively rapid convergence towards a poor quality local optimum). In some conditions, they weighted the velocity and, in some conditions, the particle's location, by some random value, thereby obtaining a sort of "dissipative particle swarm."

In bare-bones formulations of PSO, Kennedy (2003) proposed to move particles according to a probability distribution rather than through the addition of velocity—a "velocity-free" PSO. Bare-bones seeks to throw light on the relative importance of particle motion and the neighborhood topology.
Gaussian bare-bones works quite well, imitating the performance of PSO on some problems, but proving less effective on others (see also the comparisons in Richer. and Blackwell 2006). On closer examination, what appears to be a bell curve actually has a kurtosis which increases with iteration (Kennedy 2004), and the distribution has fatter tails than Gaussian. It has been suggested that the origin of this lies in the production of "bursts of outliers" (Kennedy 2004).5 The trigger for these bursts is unknown; however Kennedy discovered that if burst events are added by hand to Gaussian bare-bones, performance is improved. The conjecture, therefore, is that the fat tails in the position distribution of canonical PSO enhance the ability of the swarm to move from sub-optimal locations.
Following on from this result, Richer. and Blackwell (2006) replaced the Gaussian distribution on bare-bones with a Lévy distribution. The Lévy distribution is bell-shaped like the Gaussian but with fatter tails. The Lévy has a tunable parameter, α, which interpolates between the Cauchy distribution (α = 1) and Gaussian (α = 2). This parameter can be used to control the fatness of the tails. In a series of trials, Richer and Blackwell found that Lévy bare-bones at α = 1.4 reproduces canonical PSO behavior, a result which supports the above conjecture.

A statistical distribution also appears in canonical PSO; the p − x terms are multiplied by a random number from a uniform distribution. This injection of noise is believed to be critical to the search properties of PSO. The uniform distributed spring constant was replaced by a Gaussian random variable in (Secrest and Lamont 2003) and by the Lévy distribution in (Richer. and Blackwell 2006). The Gaussian spring constants PSO performed worse than standard PSO in a nine-function test suite, but Lévy spring constants PSO produced excellent results (Richer. and Blackwell 2006). The explanation might lie at the tails again, where large spring constants induce big accelerations and move particles away from local optima.

3. Conclusion:

In this work we have presented a review on filter design strategies using optimization techniques. DAPSO is an improved particle swarm optimization (PSO) that proposes a new definition for the velocity vector and swarm updating and hence the solution quality is improved. The distance from each particle to the global best position is calculated in order to adjust the velocity suitably of each particle. The inertia weight has been modified in this PSO to enhance its search capability that leads to a higher probability of obtaining the global optimal solution. The key feature of the modified inertia weight mechanism is to monitor the weights of particles, which linearly decrease in general applications.

References:

[1] Agrafiotis, D. K., & Cedeño, W. (2002). Feature selection for structure-activity correlation using binary particle swarms. Journal of Medicinal Chemistry, 45(5), 1098–1107.
[2] Angeline, P. (1998). Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences. In V. W. Porto, N. Saravanan, D. Waagen, & A. E. Eiben (Eds.), Proceedings of evolutionary programming VII (pp. 601–610). Berlin: Springer.
[3] L. J. Fogel, A. J. Owens, and M. J. Walsh. Artificial intelligence through a simulation of evolution. In M. Maxfield, A. Callahan, and L. J. Fogel, editors, Biophysics and Cybernetic Systems: Proc. of the 2nd Cybernetic Sciences Symposium, pp. 131–155. Spartan Books, 1965.
[4] Y. Fukuyama, S. Takayama, Y. Nakanishi, and H. Yoshida. A particle swarm optimization for reactive power and voltage control in electric power systems. In W. Banzhaf, J. Daida, A. E. Eiben, M. H. Garzon, V. Honavar, M. Jakiela,

and R. E. Smith, editors, Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1523–1528. Morgan Kaufmann Publishers, 1999.
[5] D. Gies and Y. Rahmat-Samii. Particle swarm optimization for reconfigurable phase-differentiated array design. Microwave and Optical Technology Letters, Vol. 38, No. 3, pp. 168–175, 2003.
[6] J. H. Holland. Adpatation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor, MI, 1975.
[7] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In Proceedings of the 1995 IEEE International Conference on Neural Networks, Vol. 4, pp. 1942–1948. IEEE Press, 1995.
[8] I. Rechenberg. Evolution strategy: Optimization of technical systems by means of biological evolution. Fromman-Holzboog, 1973.
[9] Blackwell, T. M. (2007). Particle swarm optimization in dynamic environments. In S. Yand, Y. Ong, & Y. Jin (Eds.), Evolutionary computation in dynamic environments (pp. 29–49). Springer, Berlin. DOI 10.1007/978-3-540-49774-5-2.
[10] Blackwell, T., & Bentley, P. J. (2002). Don't push me! Collision-avoiding swarms. In Proceedings of the IEEE congress on evolutionary computation (CEC) (pp. 1691–1696), Honolulu, HI. Piscataway: IEEE.
[11] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, International Computer Science Institute, Berkley, 1995.
[12] R. Thomsen. Flexible ligand docking using differential evolution. In Proceedings of the 2003 Congress on Evolutionary Computation, Vol. 4, pp. 2354–2361. IEEE Press, 2003.
[13] Blackwell, T. M., & Branke, J. (2006). Multi-swarms, exclusion and anti-convergence on dynamic environments. IEEE Transactions on Evolutionary Computation, 10, 459–472.
[14] R. K. Ursem and P. Vadstrup. Parameter identification of induction motors using differential evolution. In Proceedings of the 2003 Congress on Evolutionary Computation, Vol. 2, pp. 790–796. IEEE Press, 2003.
[15] Carlisle, A., & Dozier, G. (2000). Adapting particle swarm optimization to dynamic environments. In Proceedings of international conference on artificial intelligence (pp. 429–434), Las Vegas, NE.
[16] Carlisle, A., & Dozier, G. (2001). Tracking changing extrema with particle swarm optimizer. Auburn University Technical Report CSSE01-08.
[17] Clerc, M. (2004). 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.
[18] Clerc, M. (2006b). Particle swarm optimization. London: ISTE.
[19] Dorigo, M., & Stützle, T. (2004). Ant colony optimization. Cambridge: MIT Press.
[20] Eberhart, R. C., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In Proceedings of the sixth international symposium on micro machine and human science (pp. 39–43), Nagoya, Japan. Piscataway: IEEE.
[21] Eberhart, R. C., & Shi, Y. (2001). Tracking and optimizing dynamic systems with particle swarms. In Proceedings of the IEEE congress on evolutionary computation (CEC) (pp. 94–100), Seoul, Korea. Piscataway: IEEE.
[22] Eberhart, R. C., Simpson, P. K., & Dobbins, R. W. (1996). Computational intelligence PC tools. Boston: Academic Press.
[23] Hendtlass, T. (2001). A combined swarm differential evolution algorithm for optimization problems. In L. Monostori, J. Váncza & M. Ali (Eds.), Lecture notes in computer science: Vol. 2070. Proceedings of the 14th international conference on industrial and engineering applications of artificial intelligence and expert systems (IEA/AIE) (pp. 11–18), Budapest, Hungary. Berlin: Springer.
[24] Heppner, H., & Grenander, U. (1990). A stochastic non-linear model for coordinated bird flocks. In S. Krasner (Ed.), The ubiquity of chaos (pp. 233–238). Washington: AAAS.
[25] Holden, N., & Freitas, A. A. (2005). A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data. In Proceedings of the IEEE swarm intelligence symposium (SIS) (pp. 100– 107). Piscataway: IEEE.
[26] Hu, X., & Eberhart, R. C. (2001). Tracking dynamic systems with PSO: where's the cheese? In Proceedings of the workshop on particle swarm optimization. Purdue school of engineering and technology, Indianapolis, IN.
[27] Hu, X., & Eberhart, R. C. (2002). Adaptive particle swarm optimization: detection and response to dynamic systems. In Proceedings of the IEEE congress on evolutionary computation (CEC) (pp. 1666–1670), Honolulu, HI. Piscataway: IEEE.
[28] Iqbal, M., & Montes de Oca, M. A. (2006). An estimation of distribution particle swarm optimization algorithm. In M. Dorigo, L. M. Gambardella, M. Birattari, A. Martinoli, R. Poli & T. Stützle (Eds.), Lecture notes in computer science: Vol. 4150. Proceedings of the fifth international workshop on ant colony optimization and swarm intelligence ANTS 2006 (pp. 72–83). Berlin: Springer.
[29] Janson, S., & Middendorf, M. (2004). A hierarchical particle swarm optimizer for dynamic optimization problems. In G. R. Raidl (Ed.), Lecture notes in computer science: Vol. 3005. Proceedings of evoworkshops 2004: 1st European workshop on evolutionary algorithms in stochastic and dynamic environments (pp. 513–524), Coimbra, Portugal. Berlin: Springer.