Studying the Performance of Unified Particle Swarm Optimization on the Single Machine Total Weighted Tardiness Problem

K.E. Parsopoulos[1,2] and M.N. Vrahatis[1,2]

[1] Computational Intelligence Laboratory (CI Lab), Department of Mathematics, University of Patras, GR–26110 Patras, Greece
{kostasp, vrahatis}@math.upatras.gr
[2] University of Patras Artificial Intelligence Research Center (UPAIRC), University of Patras, GR–26110 Patras, Greece

Abstract. Swarm Intelligence algorithms have proved to be very effective in solving problems on many aspects of Artificial Intelligence. This paper constitutes a first study of the recently proposed Unified Particle Swarm Optimization algorithm on scheduling problems. More specifically, the Single Machine Total Weighted Tardiness problem is considered, and tackled through a scheme that combines Unified Particle Swarm Optimization and the Smallest Position Value technique for the derivation of job sequences from real–valued particles. Experiments on well–known benchmark problems are conducted with promising results, which are reported and discussed.

1 Introduction

The allocation of resources to tasks is a problem that arises very often in real–world applications. In general, problems of this type are characterized as *scheduling* problems and they are NP–hard [1, 2, 3]. The main goal in scheduling problems is the assignment of jobs (tasks) to a single or many machines such that some criteria that involve the minimization of a single or many objective functions are met.

The *Single Machine Total Weighted Tardiness* (SMTWT) problem is an NP–hard scheduling problem [1]. In SMTWT, a number, n, of jobs have to be sequentially processed on a single machine. Each job, $j = 1, 2, \ldots, n$, has a processing time, p_j, a due date, d_j, by which it should be completed, and a weight, w_j. All jobs are assumed to be available for processing at time zero. If C_j denotes the completion time of job j in a job sequence, then the tardiness of job j is defined as:

$$T_j = \max\{0, C_j - d_j\}.$$

The main goal in SMTWT problems is to find a job sequence that minimizes the sum of the weighted tardiness:

$$T = \sum_{j=1}^{n} w_j T_j. \tag{1}$$

A. Sattar and B.H. Kang (Eds.): AI 2006, LNAI 4304, pp. 760–769, 2006.

Instances of the SMTWT problem with large number of jobs cannot be solved to optimality with traditional branch–and–bound algorithms [4]. To this end, different heuristics such as Earliest Due Date and Apparent Urgency, as well as optimization algorithms such as Simulated Annealing, Tabu Search, Genetic Algorithms and Ant Colony Optimization have been successfully applied for tackling the SMTWT problem [5, 6, 7].

Recently, Particle Swarm Optimization (PSO) was applied on task assignment problems [8], as well as on the SMTWT problem [9] with promising results. In the latter case, the Smallest Position Value (SPV) representation technique was developed to transform a real–valued point to a sequence of jobs. Also, the Variable Neighborhood Search (VNS) technique [10] was employed and proved to enhance significantly the performance of PSO [9]. Unified Particle Swarm Optimization (UPSO) was recently introduced as a unified PSO scheme that combines the exploration and exploitation properties of different PSO variants [11]. Preliminary results on different problems indicate the superiority of UPSO against standard PSO variants [12, 13, 14, 15].

This paper constitutes a first investigation of UPSO on the SMTWT problem. The SPV representation scheme is adopted in our study for the derivation of job sequences from real–valued vectors. Our primary intention in this preliminary study was to assess the performance of UPSO itself and compare it with standard PSO schemes. Therefore, in order to avoid possible affection of the algorithms' dynamic, techniques like VNS that proved to significantly enhance the performance of the algorithms are not considered.

The rest of the paper is organized as follows. PSO and UPSO are briefly described in Section 2 along with the SPV representation scheme. Experimental results are reported and discussed in Section 3, and the paper concludes in Section 4.

2 Background Material

The emergent (collective) behaviors observed in natural systems have attracted a lot of attention the late years by computer scientists. Swarms of insects and animal flocks that consist of members with very limited space of actions can produce more complex behaviors as a collective, providing inspiration for the development of algorithms that can tackle NP-hard problems effectively. *Swarm Intelligence* is a subject of Artificial Intelligence that investigates the collective behavior in decentralized, self–organized systems, and promotes the development of population–based, adaptive optimization algorithms that are characterized by stochasticity, noise–tolerance and minimum requirements regarding the form of the objective function (differentiability, continuity etc.) [16].

PSO is a Swarm Intelligence algorithm introduced in 1995 by Eberhart and Kennedy [17]. The inspiration behind its development lies on the emergent behavior and information exchange in socially organized colonies of simple agents [16]. PSO was primarily used in numerical optimization tasks. However, a plethora of applications have been developed and reported in the relative literature

up–to–date [16, 18, 19, 20]. For completeness purposes, the following subsections are devoted to the description of PSO, UPSO and SPV.

2.1 Particle Swarm Optimization

PSO is a population–based algorithm. It employs a population, called a *swarm*, of search points, called *particles*, to probe the search space. Assuming an n–dimensional optimization problem,

$$\min_{x \in \mathcal{S}} f(x), \quad \mathcal{S} \subset \mathbb{R}^n,$$

then the particles are n–dimensional vectors,

$$x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^\top, \quad i = 1, \ldots, N,$$

that constitute a swarm, $\mathbb{S} = \{x_1, \ldots, x_N\}$. The numbering of particles in $\mathbb{S}$ is arbitrary. Each particle moves in the search space with an adaptable velocity,

$$v_i = (v_{i1}, v_{i2}, \ldots, v_{in})^\top,$$

and stores the best position,

$$p_i = (p_{i1}, p_{i2}, \ldots, p_{in})^\top \in \mathcal{S},$$

it has ever visited, i.e., the position with the lowest function value so far.

The computational strength of Swarm Intelligence algorithms originates from the interaction of agents that constitute the swarm, either with their environment (through stigmergy) or directly among them through information exchange. This attribute gives rise to the concept of *neighborhood*, which determines the immediate "social" environment of the agent. In PSO's framework, each particle is considered to have a neighborhood consisting of a number of other particles. These particles influence its movement with their best experience (i.e., the best positions they have discovered). However, the neighborhood is not defined directly in the search space by using a distance measure among particles, but rather in the space of the particles' indices, in order to promote the algorithm's ability for global search and avoid the computational burden of computing distances among all particles at each iteration of the algorithm.

Different neighborhood topologies have been proposed and applied in the literature with promising results [21, 22, 23]. The most common neighborhood topology is the *ring* topology, where the immediate neighbors of the particle x_i are the particles x_{i-1} and x_{i+1}, while x_1 is considered to be the particle that follows immediately after x_N. Thus, a neighborhood of radius ρ of x_i consists of the particles $x_{i-\rho}, \ldots, x_i, \ldots, x_{i+\rho}$. The ring topology is the neighborhood scheme that we adopted in our study. There are two main variants of PSO with respect to the size of neighborhood. In the *global* variant, the whole swarm is considered as the neighborhood of each particle, while, in the *local* variant, strictly smaller neighborhoods are used.

Let g_i be the index of the best particle in the neighborhood of x_i, i.e., the index of the particle that attained the best position among all the particles of the neighborhood. Then, the position of x_i is updated according to the equations [24]

$$v_i^{(k+1)} = \chi \left[v_i^{(k)} + \varphi_1 \left(p_i^{(k)} - x_i^{(k)} \right) + \varphi_2 \left(p_{g_i}^{(k)} - x_i^{(k)} \right) \right], \tag{2}$$

$$x_i^{(k+1)} = x_i^{(k)} + v_i^{(k+1)}, \tag{3}$$

where $i = 1, \ldots, N$; k is the iteration counter; χ is a parameter called *constriction coefficient* that controls the velocity's magnitude; $\varphi_1 = c_1 r_1$ and $\varphi_2 = c_2 r_2$, where c_1 and c_2 are positive acceleration parameters, called *cognitive* and *social* parameter, respectively, and r_1, r_2 are random vectors that consist of random values uniformly distributed in $[0, 1]$. All vector operations in Eqs. (2) and (3) are performed componentwise. A stability analysis of PSO, as well as recommendations regarding the selection of its parameters are provided in [24, 25].

The performance of an optimization algorithm depends heavily on the balance between its exploration and exploitation ability. In the global variant of PSO, all particles are attracted by the same best position, converging faster towards specific locations in the search space. Thus, it has better exploitation abilities, in contrast to the local variant where the information of the best position of each neighborhood is communicated slowly to the other particles of the swarm through their neighbors in the ring topology, thereby promoting exploration.

2.2 Unified Particle Swarm Optimization

UPSO has been recently proposed as a unified scheme that harnesses the local and global PSO variants, combining their exploration and exploitation capabilities [12, 13, 14, 15]. Let $\mathcal{G}_i^{(k+1)}$ denote the velocity update of the particle x_i in the global PSO variant and let $\mathcal{L}_i^{(k+1)}$ denote the corresponding velocity update for the local variant. Then, according to Eq. (2), we obtain:

$$\mathcal{G}_i^{(k+1)} = \chi \left[v_i^{(k)} + \varphi_1 \left(p_i^{(k)} - x_i^{(k)} \right) + \varphi_2 \left(p_g^{(k)} - x_i^{(k)} \right) \right], \tag{4}$$

$$\mathcal{L}_i^{(k+1)} = \chi \left[v_i^{(k)} + \varphi_1' \left(p_i^{(k)} - x_i^{(k)} \right) + \varphi_2' \left(p_{g_i}^{(k)} - x_i^{(k)} \right) \right], \tag{5}$$

where k denotes the iteration number; g is the index of the best particle of the whole swarm (global variant); and g_i is the index of the best particle in the neighborhood of x_i (local variant). The search directions $\mathcal{G}_i^{(k+1)}$ and $\mathcal{L}_i^{(k+1)}$ are combined in a single equation, resulting in the main UPSO scheme [11]:

$$\mathcal{U}_i^{(k+1)} = u\, \mathcal{G}_i^{(k+1)} + (1 - u)\, \mathcal{L}_i^{(k+1)}, \tag{6}$$

$$x_i^{(k+1)} = x_i^{(k)} + \mathcal{U}_i^{(k+1)}, \tag{7}$$

where $u \in [0, 1]$ is called the *unification factor* and it determines the influence of the global and local search direction in Eq. (6). The standard local and global PSO variant is obtained for $u = 0$ and $u = 1$, respectively. All intermediate

Table 1. An example of the Smallest Position Value representation scheme

Jobs	j	1	2	3	4	5
Particle	x_{ij}	1.45	−3.54	2.67	−2.29	−4.02
Sequence	s_{ij}	5	2	4	1	3

values of $u \in (0, 1)$ correspond to composite variants of PSO that combine the exploration and exploitation characteristics of the global and local variant.

UPSO can be further enhanced by incorporating a stochastic parameter in Eq. (6). This parameter imitates mutation in evolutionary algorithms, although, it is directed towards a direction that is consistent with the PSO dynamic. Thus, Eq. (6) can be written either as:

$$\mathcal{U}_i^{(k+1)} = r_3\, u\, \mathcal{G}_i^{(k+1)} + (1-u)\, \mathcal{L}_i^{(k+1)}, \tag{8}$$

which is mostly based on the local variant or, alternatively,

$$\mathcal{U}_i^{(k+1)} = u\, \mathcal{G}_i^{(k+1)} + r_3\, (1-u)\, \mathcal{L}_i^{(k+1)}, \tag{9}$$

which is mostly based on the global variant. The parameter $r_3 \sim \mathcal{N}(M, \Sigma)$ is a normally distributed parameter with mean vector M and variance matrix Σ. Based on the analysis of Matyas [26] for stochastic optimization algorithms, convergence in probability was proved for the schemes of Eqs. (8) and (9) [11].

2.3 The Smallest Position Value Representation Scheme

PSO and UPSO were designed to work primarily on real–valued search spaces. Thus, in different problems, proper representation schemes may be required. For example, a rounding scheme was used for the transformation of real to integer values in discrete search spaces [27, 12].

In SMTWT, each real–valued particle must correspond to a permutation of jobs. For this purpose, the SPV scheme [9] was used. More specifically, let n be the number of jobs. Then, the ith particle, $x_i = (x_{i1}, x_{i2}, \ldots, x_{in})^\top$, is n–dimensional and each component corresponds to one job, i.e., x_{ij} corresponds to the jth job. The sequence of jobs that corresponds to x_i is an integer vector

$$s_i = (s_{i1}, s_{i2}, \ldots, s_{in})^\top,$$

where s_{ij}, $j = 1, 2, \ldots, n$, is the assignment of job j in the processing order. The determination of s_{ij} is based on x_{ij} such that jobs with smaller values of x_{ij} are scheduled first. An example is provided in Table 1 for the particle $x_i = (1.45, -3.54, 2.67, -2.29, -4.02)^\top$. The smallest component of x_i is -4.02, which corresponds to the fifth job. Thus, job 5 is scheduled first and the first component of the sequence s_i takes the value $s_{i1} = 5$. The second smallest

component of x_i is -3.54, which corresponds to job 2. Therefore, job 2 is the second job in the ordering, i.e., $s_{i2} = 2$, etc.

Thus, each particle of the swarm corresponds to a sequence of jobs that is used for the computation of the total tardiness using Eq. (1). Obviously, the same sequence vector corresponds to all particles with the same ordering in their components. This property could lead the algorithm to search stagnation if some assumptions that are usually made in static optimization problems and concern the bounding of particles and velocities within specific bounds are not abandoned in the case of SMTWT. The SPV representation scheme was applied successfully with the inertia weight version of PSO [28] on the SMTWT problem [9]. We adopted SPV also in our approach, with the constriction coefficient version of PSO and UPSO.

3 Results and Discussion

We investigated the performance of three UPSO variants that proved to be very efficient in static and dynamic optimization problems [12, 13, 14, 15]. More specifically, we used the main UPSO scheme of Eq. (6) with $u = 0.2$ and $u = 0.5$, as well as the scheme with mutation defined by Eq. (8) with $u = 0.1$; $r_3 \sim \mathcal{N}(M, \Sigma)$; $M = (0, \ldots, 0)^\top$; $\Sigma = \sigma^2 I$ with $\sigma = 0.01$, and I being the $n \times n$ identity matrix. We denote these variants as UPSO_1, UPSO_2, and UPSO_m, respectively. Their performance was compared with the performance of the global and local PSO variant. These variants are denoted as PSO_g and PSO_ℓ and they are obtained from the UPSO scheme for $u = 1$ and $u = 0$, respectively. For all algorithms, the default PSO parameter set, i.e., $\chi = 0.729$ and $c_1 = c_2 = 2.05$, was used [24], while the neighborhood radius was $\rho = 1$.

The established sets of randomly generated benchmark problems of 40 and 50 jobs, each containing 125 instances, that are provided via ORLIB [7, 9] were employed in our experiments. The swarm size was equal to $N = 10 \times n$, where n is the number of jobs (also equal to the dimension of the problem using the SPV representation scheme). For each problem instance, 25 independent experiments were conducted. At each experiment, the algorithm was applied until the optimal solution was detected or a maximum number of 2000 iterations was reached. The particles and velocities were initialized randomly in $[-1, 1]^n$, although no constraints were posed on them during the execution of the algorithm.

For each algorithm, we recorded the percentage, n_{opt}, of successful experiments, i.e., experiments where the optimal solution was found, as well as the expected number of iterations, which is defined as the mean number of iterations required in the successful experiments. Also, we computed the *average relative percent deviation* from the optimal solution, which is defined as:

$$\Delta_{\text{avg}} = \sum_{i=1}^{R} \left[\frac{1}{R} \left(\frac{100(s_b^i - s^*)}{s^*} \right) \right],$$

where s^* is the optimal solution; s_b^i is the best solution obtained in the ith experiment; and R is the total number of experiments for all instances per problem,

Table 2. Results for the SMTWT problems

# Jobs	Alg.	n_{opt}	Exp. It.	Δ_{avg}	Δ_{std}
40	PSO_g	56.3%	229.6	1.692	8.931
	PSO_ℓ	47.0%	613.0	0.754	4.415
	UPSO_1	62.5%	158.1	1.765	10.865
	UPSO_2	54.1%	131.1	2.147	11.053
	UPSO_m	65.8%	525.6	0.478	3.206
50	PSO_g	42.2%	275.6	1.778	7.566
	PSO_ℓ	27.0%	338.4	1.292	3.898
	UPSO_1	43.7%	182.3	1.483	6.954
	UPSO_2	37.7%	178.4	2.329	9.352
	UPSO_m	39.6%	472.3	0.720	3.029

i.e., $R = 25 \times 125 = 3125$. Furthermore, the standard deviation, Δ_{std}, of the relative percent deviation from the optimal solution was recorded. The values Δ_{avg} and Δ_{std} provide also an intuition regarding the behavior of the algorithm in cases where it failed to detect the optimal solution, since smaller values reveal a tendency of the algorithm to converge towards sub–optimal solutions that lie closer to the optimal one. All results are reported in Table 2. The values of Δ_{avg} and scaled n_{opt} in $[0, 1]$, per algorithm for the problems of 40 and 50 jobs are depicted also in the bar graphs of Figs. 1 and 2, respectively.

In the case of 40 jobs, UPSO_m had the best performance with respect to the number of successes, followed closely by UPSO_1 ($u = 0.2$). Also, UPSO_m exhibited the best values of Δ_{avg} and Δ_{std}. This is an indication of the good quality of sub–optimal solutions it detected in the unsuccessful experiments. However, this comes with a higher computational cost, since UPSO_m required a large number of iterations. Also, we notice that PSO_g performs better than PSO_ℓ in terms of the number of successful experiments, although, it has higher values of Δ_{avg} and Δ_{std}, i.e., it is less robust. This is due to the higher exploitation ability of PSO_g compared to PSO_ℓ, which results in faster convergence but with the risk of premature convergence. UPSO_1 increases the exploitation ability of PSO_ℓ, with an immediate impact on its performance and success rates, justifying the usefulness of the unified scheme.

Similar comments can be made for 50 jobs problem. In this case, UPSO_1 outperformed all other methods with respect to the number of successes, followed by PSO_g. UPSO_m has a slightly worst performance, although with significantly smaller values of Δ_{avg}, but higher expected number of iterations. The most balanced scheme, UPSO_2 ($u = 0.5$), had better performance than PSO_ℓ, but it was always outperformed by the rest variants, similarly to the case of 40 jobs.

Summarizing the results, UPSO_1, which constitutes a modified version of PSO_ℓ with increased exploitation ability, is the most promising scheme since it

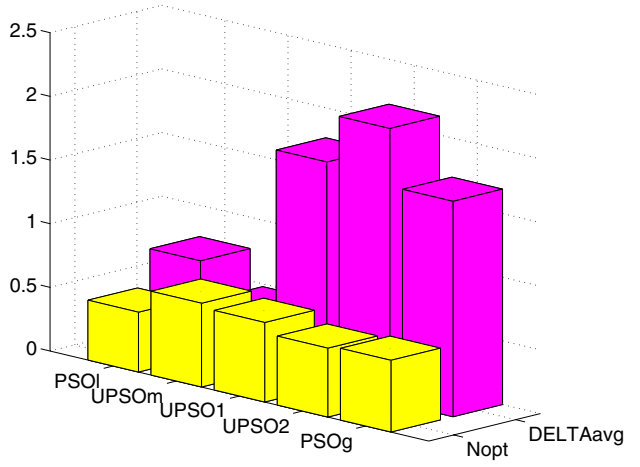

Fig. 1. Bar graph of n_{opt} normalized in $[0, 1]$, and Δ_{avg} for the 40 jobs problems

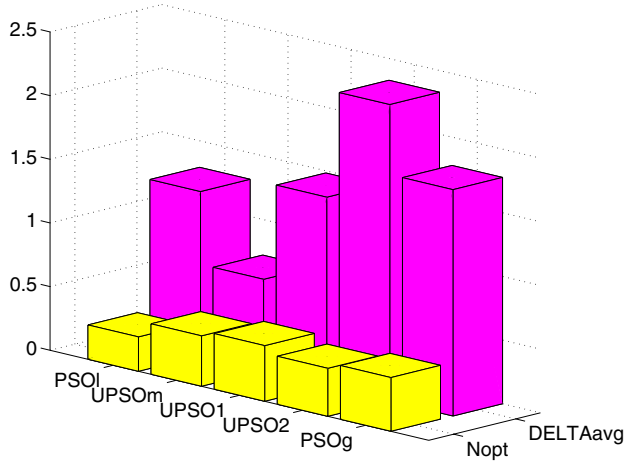

Fig. 2. Bar graph of n_{opt} normalized in $[0, 1]$, and Δ_{avg} for the 50 jobs problems

always exhibited robust behavior and required significantly smaller number of iterations than its competitive variants, UPSO$_m$ and PSO$_g$. This justifies that the unified scheme can produce more efficient PSO variants.

4 Conclusions

A first study of UPSO on the Single Machine Total Weighted Tardiness problem was provided. Widely used benchmark problems were employed and results were reported and compared with that of established PSO variants. UPSO had the best performance and robust behavior, enhancing the standard PSO variants and

justifying the usefulness of the unified scheme. Further investigation is needed to fully reveal the ability of UPSO in tackling scheduling problems, as well as possible impact of the representation scheme on the algorithm's performance.

References

1. Pinedo, M.: Scheduling: Theory, Algorithms and Systems. Prentice Hall, Englewood Cliffs (1995)
2. Johnson, D.S., Garey, M.R.: Computers and Intractability: A Guide to the Theory of NP–Completeness. W.H. Freeman & Co., Englewood Cliffs (1979)
3. Lenstra, J.K., Rinnooy Kan, H.G., Brucker, P.: Complexity of machine scheduling problem. In Studies in Integer Programming. Volume 1 of Annals of Discrete Mathematics. North–Holland, Amsterdam (1977) 343–362
4. Abdul-Razaq, T.S., Potts, C.N., Van Wassenhove, L.N.: A survey of algorithms for the single machine total weighted tardiness scheduling problem. Discrete Applied Mathematics **26** (1990) 235–253
5. Potts, C.N., Van Wassenhove, L.N.: Single machine tardiness sequencing heuristics. IIE Transactions **23** (1991) 346–354
6. Crauwels, H.A.J., Potts, C.N., Van Wassenhove, L.N.: Local search heuristics for the single machine total weighted tardiness scheduling problem. INFORMS Journal on Computing **10**(3) (1998) 341–350
7. den Besten, M., Stützle, T., Dorigo, M.: Ant colony optimization for the total weighted tardiness problem. In: Lecture Notes in Computer Science. Volume 1917. Springer-Verlag (2000) 611–620
8. Saldam, A., Ahmad, I., Al-Madani, S.: Particle swarm optimization for task assignment problem. Microprocessors and Microsystems **26** (2002) 363–371
9. Fatih Tasgetiren, M., Sevkli, M., Liang, Y.C., Gencyilmaz, G.: Particle swarm optimization algorithm for single machine total weighted tardiness problem. In: Proc. 2004 IEEE Congress on Evolutionary Computation. (2004) 1412–1419
10. Mladenovic, N., Hansen, P.: Variable neighborhood search. Computers and Operations Research **24** (1997) 563–571
11. Parsopoulos, K.E., Vrahatis, M.N.: UPSO: A unified particle swarm optimization scheme. In: Lecture Series on Computer and Computational Sciences, Vol. 1, Proc. Int. Conf. Comput. Meth. Sci. Engin. (ICCMSE 2004), VSP International Science Publishers, Zeist, The Netherlands (2004) 868–873
12. Parsopoulos, K.E., Vrahatis, M.N.: Unified particle swarm optimization for tackling operations research problems. In: Proc. IEEE 2005 Swarm Intelligence Symposium, Pasadena (CA), USA (2005) 53–59
13. Parsopoulos, K.E., Vrahatis, M.N.: Unified particle swarm optimization in dynamic environments. In Lecture Notes in Computer Science (LNCS). Volume 3449. Springer Verlag (2005) 590–599
14. Parsopoulos, K.E., Vrahatis, M.N.: Unified particle swarm optimization for solving constrained engineering optimization problems. In Lecture Notes in Computer Science (LNCS). Volume 3612. Springer Verlag (2005) 582–591
15. Parsopoulos, K.E., Vrahatis, M.N.: Parameter selection and adaptation in unified particle swarm optimization. (Mathematical and Computer Modelling) to appear.
16. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann Publishers (2001)

17. Kennedy, J., Eberhart, R.C.: Particle swarm optimization. In: Proc. IEEE Int. Conf. Neural Networks. Volume IV., IEEE Service Center (1995) 1942–1948
18. Bonabeau, E., Dorigo, M., Théraulaz, G.: Swarm Intelligence: From Natural to Artificial Swarm Intelligence. Oxford University Press, New York (1999)
19. Bonabeau, E., Meyer, C.: Swarm intelligence: A whole new way to think about business. Harvard Business Review **79**(5) (2001) 106–114
20. Engelbrecht, A.P.: Fundamentals of Computational Swarm Intelligence. Wiley (2006)
21. Kennedy, J.: Bare bones particle swarms. In: Proc. IEEE Swarm Intelligence Symposium, Indianapolis, USA, IEEE Press (2003) 80–87
22. Mendes, R., Kennedy, J., Neves, J.: The fully informed particle swarm: Simpler, maybe better. IEEE Trans. Evol. Comput. **8**(3) (2004) 204–210
23. Li, X.: Adaptively choosing neighborhood bests using species in a particle swarm optimizer for multimodal function optimization. In LNCS, Vol. 3102, Springer (2004) 105–116
24. Clerc, M., Kennedy, J.: The particle swarm–explosion, stability, and convergence in a multidimensional complex space. IEEE Trans. Evol. Comput. **6**(1) (2002) 58–73
25. Trelea, I.C.: The particle swarm optimization algorithm: Convergence analysis and parameter selection. Information Processing Letters **85** (2003) 317–325
26. Matyas, J.: Random optimization. Automatization and Remote Control **26** (1965) 244–251
27. Parsopoulos, K.E., Vrahatis, M.N.: Recent approaches to global optimization problems through particle swarm optimization. Natural Computing **1**(2–3) (2002) 235–306
28. Shi, Y., Eberhart, R.C.: A modified particle swarm optimizer. In: Proc. IEEE CEC 1998, IEEE Service Center (1998) 69–73