Handling Constraints in Particle Swarm Optimization using a Small Population Size

Juan C. Fuentes Cabrera and Carlos A. Coello Coello

CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Computación
Av. IPN No. 2508, San Pedro Zacatenco
México D.F. 07360, MÉXICO
jfuentes@computacion.cs.cinvestav.mx
ccoello@cs.cinvestav.mx

Abstract. This paper presents a particle swarm optimizer for solving constrained optimization problems which adopts a very small population size (five particles). The proposed approach uses a reinitialization process for preserving diversity, and does not use a penalty function nor it requires feasible solutions in the initial population. The leader selection scheme adopted is based on the distance of a solution to the feasible region. In addition, a mutation operator is incorporated to improve the exploratory capabilities of the algorithm. The approach is tested with a well-know benchmark commonly adopted to validate constraint-handling approaches for evolutionary algorithms. The results show that the proposed algorithm is competitive with respect to state-of-the-art constraint-handling techniques. The number of fitness function evaluations that the proposed approach requires is almost the same (or lower) than the number required by the techniques of the state-of-the-art in the area.

1 Introduction

A wide variety of Evolutionary Algorithms (EAs) have been used to solve different types of optimization problems. However, EAs are unconstrained search techniques and thus require an additional mechanism to incorporate the constraints of a problem into their fitness function [3]. Penalty functions are the most commonly adopted approach to incorporate constraints into an EA. However, penalty functions have several drawbacks, from which the most important are that they require a fine-tuning of the penalty factors (which are problem-dependent), since both under- and over-penalizations may result in an unsuccessful optimization [3].

Particle swarm optimization (PSO) is a population based optimization technique inspired on the movements of a flock of birds or fish. PSO has been successfully applied in a wide of variety of optimization tasks in which it has shown a high convergence rate [10].

This paper is organized as follows. In section 2, we define the problem of our interest and we introduce the Particle Swarm Optimization algorithm. The previous related work is provided in Section 3. Section 4 describes our approach including the reinitialization process, the constraint-handling mechanism and the mutation operator adopted. In Section 5, we present our experimental setup and the results obtained. Finally, Section 6 presents our conclusions and some possible paths for future research.

2 Basic Concepts

We are interested in the general nonlinear programming problem in which we want to:

$$\textit{Find } \overrightarrow{x} \textit{ which optimizes } f(\overrightarrow{x}) \tag{1}$$

subject to:

$$g_i(\overrightarrow{x}) \leq 0, i = 1, ..., n \tag{2}$$

$$h_j(\overrightarrow{x}) = 0, j = 1, ..., p \tag{3}$$

where $\overrightarrow{x}$ is the vector of solutions $\overrightarrow{x} = [x_1, x_2, ..., x_r]^T$, n is the number of inequality constraints and p is the number of equality constraints (in both cases, constraints could be linear or nonlinear). If we denote with F to the feasible region (set of points which satisfy the inequality and equality constraints) and with S to the search space then $F \subseteq S$. For an inequality constraint that satisfies $g_i(\boldsymbol{x}) = 0$, the we will say that is **active** at $\overrightarrow{x}$. All equality constraints h_j (regardless of the value of $\overrightarrow{x}$ used) are considered active at all points of F.

2.1 Particle Swarm Optimization

Our approach is based in The Particle Swarm Optimization (PSO) algorithm which was introduced by Eberhart and Kennedy in 1995 [4]. PSO is a population-based search algorithm based on the simulation of social behavior of birds within a flock [10]. In PSO, each individual (particle) of the population (swarm) adjusts its trajectory according to its own flying experience and the flying experience of the other particles within its topological neighborhood in the search space. In the PSO algorithm, the population and velocities are randomly initialized at the beginning of the search, and then they are iteratively updated, based on their previous positions and those of each particle's neighbors. Our proposed approach implements equations (4) and (5), proposed in [19] for computing the velocity and the position of a particle.

$$v_{id} = w \times v_{id} + c_1 r_1 (pb_{id} - x_{id}) + c_2 r_2 (lb_{id} - x_{id}) \tag{4}$$

$$x_{id} = x_{id} + v_{id} \tag{5}$$

where c_1 and c_2 are both positive constants, r_1 and r_2 are random numbers generated from a uniform distribution in the range [0,1], w is the inertia weight that is generated in the range (0,1].

There are two versions of the PSO algorithm: the **global** version and the **local** version. In the global version, the neighborhood consists of all the particles of the swarm and the best particle of the population is called the "global best" (*gbest*). In contrast, in the local version, the neighborhood is a subset of the population and the best particle of the neighborhood is called "local best" (*lbest*).

3 Related Work

When incorporating constraints into the fitness function of an evolutionary algorithm, it is particularly important to maintain diversity in the population and to be able to keep solutions both inside and outside the feasible region [3, 14]. Several studies have shown that, despite their popularity, traditional (external) penalty functions, even when used with dynamic penalty factors, tend to have difficulties to deal with highly constrained search spaces and with problems in which the constraints are active in the optimum [3, 11, 18]. Motivated by this fact, a number of constraint-handling techniques have been proposed for evolutionary algorithms [16, 3].

Remarkably, there has been relatively little work related to the incorporation of constraints into the PSO algorithm, despite the fact that most real-world applications have constraints. In previous work, some researchers have assumed the possibility of being able to generate in a random way feasible solutions to feed the population of a PSO algorithm [7, 8]. The problem with this approach is that it may have a very high computational cost in some cases. For example, in some of the test functions used in this paper, even the generation of one million of random points was insufficient to produce a single feasible solution. Evidently, such a high computational cost turns out to be prohibitive in real-world applications.

Other approaches are applicable only to certain types of constraints (see for example [17]). Some of them rely on relatively simple mechanisms to select a leader, based on the closeness of a particle to the feasible region, and adopt a mutation operator in order to maintain diversity during the search (see for example [20]). Other approaches rely on carefully designed topologies and diversity mechanisms (see for example [6]). However, a lot of work remains to be done regarding the use of PSO for solving constrained optimization problems. For example, the use of very small population sizes has not been addressed so far in PSO, to the authors' best knowledge, and this paper precisely aims to explore this venue in the context of constrained optimization.

4 Our Proposed Approach

As indicated before, our proposed approach is based on the local version of PSO, since there is evidence that such model is less prone to getting stuck in local minima [10]. Despite the existence of a variety of population topologies [9], we adopt a randomly generated neighborhood topology for our approach. It also uses a reinitialization process in order to maintain diversity in the population, and it adopts a mutation operator that aims to improve the exploratory capabilities of PSO (see Algorithm 1).

Since our proposed approach uses a very small population size, we called it **Micro-PSO**, since this is analogous to the micro genetic algorithms that have been in use for several years (see for example [12, 2]). Its constraint-handling mechanism, together with its reinitialization process and its mutation operator, are all described next in more detail.

Algorithm 1: Pseudocode of our proposed Micro-PSO

```
begin
    for $i = 1$ to Number of particles do
        Initialize position and velocity randomly;
        Initialize the neighborhood (randomly);
    end
    $cont = 1$;
    repeat
        if $cont$ ==reinitialization generations number then
            Reinitialization process;
            $cont = 1$;
        end
        Compute the fitness value $G(x_i)$;
        for $i = 1$ to Number of particles do
            if $G(x_i) > G(xpb_i)$ then
                for $d = 1$ to number of dimensions do
                    $xpb_{id} = x_{id}$; // $xpb_i$ is the best position so far;
                end
            end
            Select the local best position in the neighborhood $lb_i$;
            for $d = 1$ to number of dimensions do
                $w = rand()$; // random(0,1];
                $v_{id} = w \times v_{id} + c_1 r_1 (pbx_{id} - x_{id}) + c_2 r_2 (lb_{id} - x_{id})$;
                $x_{id} = x_{id} + v_{id}$;
            end
        end
        cont =cont +1;
        Perform mutation;
    until Maximum number of generations ;
    Report the best solution found.
end
```

4.1 Constraint-Handling Mechanism

We adopted the mechanism proposed in [20] for selecting leaders. This mechanism is based both on the feasible solutions and the fitness value of a particle. When two feasible particles are compared, the particle that has the highest fitness value wins. If one of the particles is infeasible and the other one is feasible, then the feasible particle wins. When two infeasible particles are compared, the particle that has the lowest fitness value wins. The idea is to select leaders that, even when could be infeasible, lie close to the feasible region.

We used equation (6) for assigning fitness to a solution:

$$fit(\overrightarrow{x}) = \begin{cases} f_i(\overrightarrow{x}) & \text{if feasible} \\ \sum_{j=1}^{n} g_j(\overrightarrow{x}) + \sum_{k=1}^{p} |h_k(\overrightarrow{x})| & \text{otherwise} \end{cases} \tag{6}$$

4.2 Reinitialization Process

The use of a small population size accelerates the loss of diversity at each iteration, and therefore, it is uncommon practice to use population sizes that are too small. However, in the genetic algorithms literature, it is known that it is possible, from a theoretical point of view, to use very small population sizes (no more than 5 individuals) if appropriate reinitialization processes are implemented [12, 2]. In this paper, we incorporate one of these reinitialization processes taken from the literature on micro genetic algorithms. Our mechanism is the following: after certain number of iterations (replacement generations), the swarm is sorted based on fitness value, but placing the feasible solutions on top. Then, we replace the *rp* particles (replacement particles) by randomly generated particles (position and velocity), but allow the *rp* particles to hold their best position (pbest). The idea of mixing evolved and randomly generated particles is to avoid premature convergence.

4.3 Mutation Operator

Although the original PSO algorithm had no mutation operator, the addition of such mutation operator is a relatively common practice nowadays. The main motivation for adding this operator is to improve the performance of PSO as an optimizer, and to improve the overall exploratory capabilities of this heuristic [1]. In our proposed approach, we implemented the mutation operator developed by Michalewicz for Genetic Algorithms [15]. It is worth noticing that this mutation operator has been used before in PSO, but in the context of unconstrained multimodal optimization [5]. This operator varies the magnitude added or substracted to a solution during the actual mutation, depending on the current iteration number (at the beginning of the search, large changes are allowed, and they become very small towards the end of the search). We apply the mutation operator in the particle's position, for all of its dimensions:

$$x_{id} = \begin{cases} x_{id} + \Delta(t, UB - x_{id}) & \text{if } R = 0 \\ x_{id} - \Delta(t, x_{id} - LB) & \text{if } R = 1 \end{cases} \tag{7}$$

where t is the current iteration number, UB is the upper bound on the value of the particle dimension, LB is the lower bound on the particle dimension value, R is a randomly generated bit (zero and one both have a 50% probability of being generated) and $\delta(t, y)$ returns a value in the range $[0, y]$. $\delta(t, y)$ is defined by:

$$\Delta(t, y) = y * (1 - r^{1-(\frac{t}{T})^b}) \quad (8)$$

where r is a random number generated from a uniform distribution in the range[0,1], T is the maximum number of iterations and b is a tunable parameter that defines the non-uniformity level of the operator. In this approach, the b parameter is set to 5 as suggested in [15].

5 Experiments and Results

For evaluating the performance of our proposed approach, we used the thirteen test functions described in [18]. These test functions contain characteristics that make them difficult to solve using evolutionary algorithms. We performed fifty independent runs for each test function and we compared our results with respect to three algorithms representative of the state-of-the-art in the area: Stochastic Ranking (SR) [18], the Simple Multimembered Evolution Strategy (SMES) [14], and the Constraint-Handling Mechanism for PSO (CHM-PSO) [20]. Stochastic Ranking uses a multimembered evolution strategy with a static penalty function and a selection based on a stochastic ranking process, in which the stochastic component allows infeasible solutions to be given priority a few times during the selection process. The idea is to balance the influence of the objective and penalty function. This approach requires a user-defined parameter called Pf wich determines this balance [18]. The Simple Multimembered Evolution Strategy (SMES) is based on a $(\mu + \lambda)$ evolution strategy, and it has three main mechanisms: a diversity mechanism, a combined recombination operator and a dynamic step size that defines the smoothness of the movements performed by the evolution strategy [14].

In all our experiments, we adopted the following parameters:

- W = random number from a uniform distribution in the range [0,1].
- C1 = C2 = 1.8.
- population size = 5 particles;
- number of generations = 48,000;
- number of replacement generations = 100;
- number of replacement particles = 2;
- mutation percent = 0.1;
- $\epsilon = 0.0001$, except for g04, g05, g06, g07, in which we used $\epsilon = 0.00001$.

The statistical results of our Micro-PSO are summarized in Table 1. Our approach was able to find the global optimum in five test functions (g02, g04, g06, g08, g09, g12) and it found solutions very close to the global optimum in the remaining test functions, with the exception of g10. We compare the population size, the tolerance value and the number of evaluations of the objective function of our approach with respect to SR, CHM-PSO, and SMES in Table 2.

When comparing our approach with respect to SR, we can see that ours found a better solution for g02 and similar results in other eleven problems, except for

Table 1. Results obtained by our Micro-PSO over 50 independent runs.

Statistical Results of our Micro-PSO						
TF	Optimal	Best	Mean	Median	Worst	St.Dev.
g01	-15.000	-15.0001	-13.2734	-13.0001	-9.7012	1.41E+00
g02	0.803619	0.803620	0.777143	0.778481	0.711603	1.91E-02
g03	1.000	1.0004	0.9936	1.0004	0.6674	4.71E-02
g04	-30665.539	-30665.5398	-30665.5397	-30665.5398	-30665.5338	6.83E-04
g05	5126.4981	5126.6467	5495.2389	5261.7675	6272.7423	4.05E+02
g06	-6961.81388	-6961.8371	-6961.8370	-6961.8371	-6961.8355	2.61E-04
g07	24.3062	24.3278	24.6996	24.6455	25.2962	2.52E-01
g08	0.095825	0.095825	0.095825	0.095825	0.095825	0.00E+00
g09	680.630	680.6307	680.6391	680.6378	680.6671	6.68E-03
g10	7049.250	7090.4524	7747.6298	7557.4314	10533.6658	5.52E+02
g11	0.750	0.7499	0.7673	0.7499	0.9925	6.00E-02
g12	1.000	1.0000	1.0000	1.0000	1.0000	0.00E+00
g13	0.05395	0.05941	0.81335	0.90953	2.44415	3.81E-01

g10, in which our approach does not perform well (see Table 3). Note however that SR performs 350,000 objective function evaluations and our approach only performs 240,000.

Table 2. Comparison of the population size, the tolerance value and the number of evaluations of the objective function of our approach with respect to SR, CHM-PSO and SMES.

Constraint-Handling Technique	Population Size	Tolerance Value (ϵ)	Number of Evaluations of the Objective Function
SR	200	0.001	350,000
CHMPSO	40	-	340,000
SMES	100	0.0004	240,000
Micro-PSO	5	0.0001	240,000

Compared with respect to CHM-PSO, our approach found a better solution for g02, g03, g04, g06, g07, g09, and g13 and the same or similar results in other five problems (except for g10) (see Table 4). Note again that CMPSO performs 340,000 objective function evaluations, and our approach performs only 240,000. It is also worth indicating that CHMOPSO is one of the best PSO-based constraint-handling methods known to date.

When comparing our proposed approach against the SMES, ours found better solutions for g02 and g09 and the same or similar results in other ten problems, except for g10 (see Table 5). Both approaches performed 240,000 objective function evaluations.

Table 3. Comparison of our approach with respect the Stochastic Ranking (SR).

		Best Result		Mean Result		Worst Result	
TF	Optimal	Micro-PSO	SR	Micro-PSO	SR	Micro-PSO	SR
g01	-15.0000	-15.0001	-15.000	-13.2734	-15.000	-9.7012	-15.000
g02	0.803619	0.803620	0.803515	0.777143	0.781975	0.711603	0.726288
g03	1.0000	1.0004	1.000	0.9936	1.000	0.6674	1.000
g04	-30665.539	-30665.539	-30665.539	-30665.539	-30665.539	-30665.534	-30665.539
g05	5126.4981	5126.6467	5126.497	5495.2389	5128.881	6272.7423	5142.472
g06	-6961.81388	-6961.8371	-6961.814	-6961.8370	-6875.940	-6961.8355	-6350.262
g07	24.3062	24.3278	24.307	24.6996	24.374	25.2962	24.642
g08	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825
g09	680.630	680.6307	680.630	680.6391	680.656	680.6671	680.763
g10	7049.250	7090.4524	7054.316	7747.6298	7559.192	10533.6658	8835.655
g11	0.750	0.7499	0.750	0.7673	0.750	0.9925	0.750
g12	1.000	1.0000	1.000	1.0000	1.000	1.0000	1.000
g13	0.05395	0.05941	0.053957	0.81335	0.067543	2.44415	0.216915

6 Conclusions and Future Work

We have proposed the use of a PSO algorithm with a very small population size (only five particles) for solving constrained optimization problems. The proposed technique adopts a constraint-handling mechanism during the selection of leaders, it performs a reinitialization process and it implements a mutation operator. The proposed approach is easy to implement, since its main additions are a sorting and a reinitialization process. The computational cost (measured in terms of the number of evaluation of the objective function) that our Micro-PSO requires is almost the same (or lower) than the number required by the techniques with respect to which it was compared, which are representative of the state-of-the-art in the area. The results obtained show that our approach is competitive and could then, be a viable alternative for using PSO for solving constrained optimization problems.

As part of our future work, we are interested in comparing our approach with the extended benchmark for constrained evolutionary optimization [13]. We are also interested in comparing our results with respect to more recent constraint-handling methods that use PSO as their search engine (see for example [6]). We will also study the sensitivity of our Micro-PSO to its parameters, aiming to find a set of parameters (or a self-adaptation mechanism) that improves its robustness (i.e., that reduces the standard deviations obtained). Finally, we are also interested in experimenting with other neighborhood topologies and other types of

Table 4. Comparison of our approach with respect the Constraint-Handling Mechanism for PSO (CHM-PSO).

		Best Result		Mean Result		Worst Result	
TF	Optimal	Micro-PSO	CHM-PSO	Micro-PSO	CHM-PSO	Micro-PSO	CHM-PSO
g01	-15.0000	-15.0001	-15.000	-13.2734	-15.000	-9.7012	-15.000
g02	0.803619	0.803620	0.803432	0.777143	0.790406	0.711603	0.755039
g03	1.000	1.0004	1.004720	0.9936	1.003814	0.6674	1.000249
g04	-30665.539	-30665.539	-30665.500	-30665.539	-30665.500	-30665.534	-30665.500
g05	5126.4981	5126.6467	5126.6400	5495.2389	5461.08133	6272.7423	6104.7500
g06	-6961.8138	-6961.837	-6961.810	-6961.837	-6961.810	-6961.835	-6961.810
g07	24.3062	24.3278	24.3511	24.6996	24.35577	25.2962	27.3168
g08	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825
g09	680.6300	680.6307	680.638	680.6391	680.85239	680.6671	681.553
g10	7049.2500	7090.4524	7057.5900	7747.6298	7560.04785	10533.6658	8104.3100
g11	0.7500	0.7499	0.7499	0.7673	0.7501	0.9925	0.75288
g12	1.0000	1.0000	1.000	1.0000	1.000	1.0000	1.000
g13	0.05395	0.05941	0.068665	0.81335	1.71642	2.44415	13.6695

Table 5. Comparison of our approach with respect the Simple Multimembered Evolution Strategy (SMES).

		Best Result		Mean Result		Worst Result	
TF	Optimal	Micro-PSO	SMES	Micro-PSO	SMES	Micro-PSO	SMES
g01	-15.0000	-15.0001	-15.000	-13.2734	-15.000	-9.7012	-15.000
g02	0.803619	0.803620	0.803601	0.777143	0.785238	0.711603	0.751322
g03	1.0000	1.0004	1.000	0.9936	1.000	0.6674	1.000
g04	-30665.5390	-30665.5398	-30665.539	-30665.5397	-30665.539	-30665.5338	-30665.539
g05	5126.4980	5126.6467	5126.599	5495.2389	5174.492	6272.7423	5304.167
g06	-6961.8140	-6961.8371	-6961.814	-6961.8370	-6961.284	-6961.8355	-6952.482
g07	24.3062	24.3278	24.327	24.6996	24.475	25.2962	24.843
g08	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825
g09	680.6300	680.6307	680.632	680.6391	680.643	680.6671	680.719
g10	7049.2500	7090.4524	7051.903	7747.6298	7253.047	10533.6658	7638.366
g11	0.7500	0.7499	0.75	0.7673	0.75	0.9925	0.75
g12	1.0000	1.0000	1.000	1.0000	1.000	1.0000	1.000
g13	0.05395	0.05941	0.053986	0.81335	0.166385	2.44415	0.468294

reinitialization processes, since they could improve our algorithm's convergence capabilities (i.e., we could reduce the number of objective function evaluations performed) as well as the quality of the results achieved.

Acknowledgements

The first author gratefully acknowledges support from CONACyT to pursue graduate studies at CINVESTAV-IPN's Computer Science Department. The second author acknowledges support from CONACyT project No. 45683-Y.

References

1. Paul S. Andrews. An investigation into mutation operators for particle swarm optimization. In *Proceedings of the 2006 IEEE Congress on Evolutionary Computation, (CEC'2006)*, pages 3789–3796, Vancouver, Canada, July 2006.
2. Carlos A. Coello Coello and Gregorio Toscano Pulido. Multiobjective optimization using a micro-genetic algorithm. In in L. Spector, E.D. Goodman, A. Wu, W. Langdon, H.M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk M., Garzon, and E. Burke, editors, *Genetic and Evolutionary Computation Conference, GECCO,2001*, pages 274–282, San Francisco, California, 2001. Morgan Kaufmann Publishers.
3. Carlos A. Coello Coello. Theoretical and numerical constraint handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11–12):1245–1287, January 2002.
4. R. Eberhart and J. Kennedy. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE Press, 1995.
5. S.C. Esquivel and C.A. Coello Coello. On the use of particle swarm optimization with multimodal functions. In *Proceedings of the 2003 IEEE Congress on Evolutionary Computation (CEC'2003)*, pages 1130–1136, Canberra, Australia, 2003. IEEE Press.
6. Arturo Hernández Aguirre, Angel E. Muñoz Zavala, E. Villa Diharce, and S. Botello Rionda. COPSO: Constrained Optimization via PSO algorithm. Technical Report I-07-04, Center of Research in Mathematics (CIMAT), Guanajuato, México, 2007.
7. Xiaohui Hu and Russell Eberhart. Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)*, volume 5. Orlando, USA, IIIS, July 2002.
8. Xiaohui Hu, Russell C. Eberhart, and Yuhui Shi. Engineering Optimization with Particle Swarm. In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pages 53–57. Indianapolis, Indiana, USA, IEEE Service Center, April 2003.
9. J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Proceedings of the 2002 IEEE Congress on Evolutionary Computation*, volume 2, pages 1671–1676, Honolulu, Hawaii, USA, May 2002. IEEE Press.
10. James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kauffmann Publishers, San Francisco, California, 2001.
11. Slawomir Koziel and Zbigniew Michalewicz. Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation*, 7(1):19–44, 1999.

12. Kalmanje Krishnakumar. Micro-genetic algorithms for stationary and non-stationary function optimization. *SPIE Proceedings: Intelligent Control and Adaptive Systems*, 1196:289–296, 1989.
13. J.J. Liang, T.P. Runarsson, E. Mezura-Montes, M. Clerc, P.N. Suganthan, C.A. Coello Coello, and K. Deb. Problem definitions and evaluation criteria for the cec 2006 special session on constrained real-parameter optimization. Technical report, Nanyang Technological University, Singapore, 2006.
14. Efrén Mezura-Montes and Carlos A. Coello Coello. A simple multimembered evolution strategy to solve constrained optimization problems. *Transactions on Evolutionary Computation*, 9(1):1–17, February 2005.
15. Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.
16. Zbigniew Michalewicz and Marc Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1):1–32, 1996.
17. Ulrich Paquet and Andries p Engelbrecht. A New Particle Swarm Optimiser for Linearly Constrained Optimization. In *Proceedings of the Congress on Evolutionary Computation 2003 (CEC'2003)*, volume 1, pages 227–233, Piscataway, New Jersey, December 2003. Canberra, Australia, IEEE Service Center.
18. T.P. Runanarsson and X. Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):248–249, September 2000.
19. Y. Shi and R.C. Eberhart. A modified particle swarm optimizer. In *Proceedings of the 1998 IEEE Congress on Evolutionary Computation*, pages 69–73. IEEE Press, 1998.
20. Gregorio Toscano Pulido and Carlos A. Coello Coello. A constraint-handling mechanism for particle swarm optimization. In *Proceedings of the 2004 IEEE Congress on Evolutionary Computation (CEC'2004)*, volume 2, pages 1396–1403, Portland, Oregon, USA, June 2004. IEEE Press.