Boundary Handling Approaches in Particle Swarm Optimization

Nikhil Padhye[1] and Kalyanmoy Deb[2]

[1] Department of Mechanical Engineering
Massachusetts Institute of Technology, Cambridge, MA-02139
npdhye@mit.edu
[2] Department of Mechanical Engineering
Indian Institute of Technology Kanpur, Kanpur-208016, U.P., India
deb@iitk.ac.in

Abstract. In recent years, Particle Swarm Optimization (PSO) methods have gained popularity for solving single objective (and other) optimization tasks in the pool of evolutionary approaches such as Genetic Algorithm (GA), Differential Evolution (DE) and Evolutionary Strategy (ES). In particular, solving constrained optimization problems using swarm methods has been attempted in past and arguably stays as one of the challenging issues, far from being fully resolved. A commonly occurring situation for constrained optimization problems is one in which constraints manifest themselves in form of variable bounds only. In such scenarios the issue of constraint-handling is somewhat simplified and requires effective *bound handling* strategies. This paper attempts to review popular *bound handling* methods, in context to PSO, and proposes two new methods which are found to be efficient and consistent in terms of performance over several simulation scenarios. The effectiveness of *bound handling* methods are shown in combination with PSO; however the methods are general and can be applied with any other optimization procedure.

1 Introduction

Optimization problems are wide-spread in domains of science, engineering, commerce, and even humanities. For such optimization problems the goal is to minimize (without any loss of generality, since all optimization problems can be formulated as minimization type) a pre-defined objective which is often expressed as a mathematical function. The objective function is dependent on several variables which define the problem itself. For a particular combination(s) of problem variables the objective function achieves an optimal value and the minimization task is achieved; the corresponding set of variables is referred to as the optimal set. However, the real-world scenarios are often not straight-forward and place certain restrictions on the problem variables thereby limiting them from taking any real value. Put in other words, the problem variables have to satisfy certain pre-defined mathematical relationships in order for the problem to remain defined. These mathematical relationships are commonly called constraints.
The most general form of constrained optimization problem (with equality/inequality constraints, and/or variable bounds) can be written as a nonlinear programming (NLP) problem as:

$$\begin{aligned} Minimize \quad & f(\bar{x}) \\ Subjectto \quad & g_j(\bar{x}) \geq 0, \quad j = 1, \dots, J \\ & h_k(\bar{x}) = 0, \quad k = 1, \dots, K \\ & x_i^l \leq x_i \leq x_i^u, \; i = 1, \dots, n \end{aligned} \tag{1}$$

The above NLP problem contains n variables (i.e. $\bar{x}$ is vector of size n), J greater-than-equal-to type inequality constraints (less-than-equal-to can be expressed in this form by multiplying both sides by a negative) and K equality type constraints. The problem variables $x_i s$ are bounded within upper and lower limits.
The classical optimization algorithms employ several constraint handling methods such as penalty function, Lagrange multiplier, complex search, cutting plane, reduced gradient, gradient projection, etc. For details see [7, **?**]. Directly or indirectly several of these methods can also be (and have been) applied with evolutionary searches.
In this paper we are interested in a special class of constrained optimization problems which have only the variable bounds, i.e. types of problems where there are no equality or inequality constraints. The motivation to study such problems arises from the fact that many real optimization problems have variable bounds as the only constraints. Therefore, it makes appropriate sense to search for efficient techniques to address this special class of problems, as opposed to employing general constraint handling techniques.
In context to Particle Swarm Optimization (PSO), several bound handling methods have already been proposed [6, 5, 1]. However, many of these past proposals exploit the information about location of the optimum and fail to perform when location of optimum changes. The major contribution of this paper is to come up with robust bound handling technique which never fails to perform. This is achieved by proposing two stochastic and adaptive distributions to bring particles back into the feasible region once they fly out the search space. The existing and proposed bound handling methods are tested on two standard test problems under different scenarios and their performance is compared based on statistical basis.
The rest of the paper is organized as follows: Section 2 reviews different bound handling techniques and provides a detailed description on existing two adaptive bound handling methods proposed in this paper. Section 3 provides a description on test problems and different experiment scenarios adopted in this study. Section 4 investigates the simulation results and conducts performance comparisons. Finally, conclusions are made in Section 5.

2 Bound Handling Mechanisms

Handling of variable bounds (by treating them as constraints) can be done in several ways. One such simple approach is to treat variable bounds as inequality type constraints and use *penalty based* approaches (commonly used in EAs). The *penalty based* approaches allow solutions to lie in the infeasible regions and degrade their "fitness" values thereby assigning less importance to infeasible solutions. The major disadvantage of this approach lies in the fact that the objective function may not be defined in the infeasible region, in which case *penalty based*

approach is useless. Even when the definition of objective can be extended beyond the variable bounds, additional parameters (like penalty parameter *'R'*) are needed from the user to compute the penalty. The inclusion of a penalty parameter may add artificial optima in the problem and misguide the EA search [2].
To avoid penalty based approaches only following two alternatives are possible: (a) creation of feasible-only solutions during the evolutionary search, or (b) an explicit mechanism to *repair* an infeasible solution i.e. bringing the infeasible solution back into the feasible search space. Generation of feasible-only solutions in case of EAs is not always straight- forward task, if not impossible. For e.g., the generation of two children from two parents in simulated binary crossover can be done by redistributing the probability distribution function in such a way that infeasible regions are assigned zero probability for child-creation [2], but, for other crossover operators such as simplex, or evolutionary methods like PSO and DE, means to create feasible-only solutions at the generational step is often not possible. In context to PSO therefore it is necessary to have an explicit scheme which can bring an infeasible particle back into the feasible search region.
In past, several methods have been proposed to bring PSO particles back into feasible regions. In this study we shall refer to them as *bound handling methods or techniques*. An exhaustive recollection and simulation for all *bound handling techniques* is beyond the scope of this chapter. Hence, we shall focus our efforts only on the most popular and representative *bound handling techniques*.
The *bound handling techniques* can be broadly divided into two groups – (A) techniques which carry out feasibility search variable wise, and (B) techniques which carry out feasibility search vectorically. According to group (A) techniques, for every solution, each variable is tested for its feasibility with respect to its bounds, and made feasible if found to violate any bound. Here, only the variables violating their bounds are altered, independently, and other variables remain unchanged until they are tested and found to violate any bounds.
According to group B techniques, if a solution (vector location) is found to violate any of the variable bounds, it is brought back into the search space along a vector direction. In such cases, the variables which have not violated any bound are also modified.
It can be speculated that for separable problems (where variables are not linked with one-another), techniques belonging to group (A) are likely to perform well. However, for problems where optimization of the function requires high-degree of correlated variable alterations, group (B) techniques may become more useful (one such method named *Adaptive Distribution* is proposed in this chapter and utilizes this fact while bringing solutions back into the feasible regions in a hopefully meaningful way). Next we discuss some popular *bound handling methods*, using which an infeasible solution (violating variable bounds) can be made feasible.

2.1 Existing Bound Handling Methods

Random This is one of the simplest strategies and belongs to group (A). One-by-one, each variable is checked for bound-violations and modified if necessary. Figure 1 shows a current infeasible location X_c along some particular dimension. Let, U and L denote the upper and lower bounds for the corresponding dimension (the same notation is used from here-on). The current location X_C, exceeds the upper bound U, and is made feasible by choosing a random location X_R in [L, U].

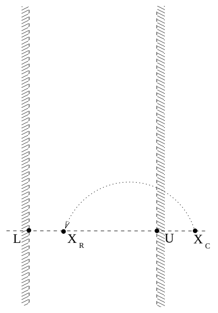

Fig. 1. Variable wise *Random* strategy for handling bounds.

The *Random* strategy does not utilize any information of a solution's past position and explicitly promotes diversity in search space. Due to such nature accuracy in convergence can be greatly affected. Overall, this strategy is applicable with any evolutionary algorithm.

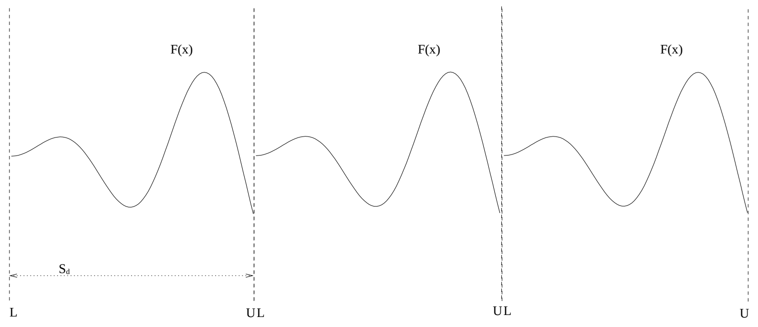

Fig. 2. Variable wise *Periodic* strategy for handling bounds.

Periodic This strategy maps an infeasible location, for each variable violating the bounds, to a feasible location by assuming an infinite search space and was originally proposed in [8]. This is done by placing repeated copies of original search space along the dimension of interest as shown Figure 2. For example, let X_C denote the current location of a solution along d^{th} dimension, then X_C is mapped to X_C^{new} as follows:

$$X_C \rightarrow X_C^{new} = \begin{cases} U - (L - X_C)\%S_d & \text{IF } X_C < \text{L} \\ L + (X_C - U)\%S_d & \text{IF } X_C > \text{U} \end{cases}$$

The *Periodic* method handles all the variables separately and allows the infeasible solution to re-enter the search space from an end which is opposite to where it left the search space. For problems where optima is at the boundary this approach is rendered ineffective, as majority of solutions approaching the boundary optima shall “fall outside” the search space and will be brought back into the search space from opposite end. The entire effort carried out in locating the optima may be lost. For unimodal problems with optima at the center of the boundary the *Periodic* approach may be useful. For PSO, two variants of this strategy are studied (depending on the way in which velocity is computed), details of which are provided later.

Set on Boundary As the name suggests, according to this strategy the individual is reset on the bound of the variable which it exceeds. The strategy belongs of group (A). For example, along d^{th} dimension, let X_C denote a current location of a solution, then X_C is set to X_C^{new} as follows:

$$X_C \rightarrow X_C^{new} = \begin{cases} L & \text{IF } X_C < \text{L} \\ U & \text{IF } X_C > \text{U} \end{cases}$$

Clearly this approach biases the infeasible solutions on the search boundaries and can be highly helpful in cases where problem optima lies on the boundary of the variables. For PSO, three variants of this strategy can be done (depending on the way in which velocity is computed). The details on the three variants are provided later.

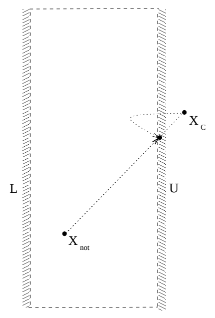

Fig. 3. Vector based *SHR.* strategy for handling bounds.

SHR Originally *Shrink(SHR.)* method was introduced in context to PSO in [1]. The goal of the *SHR.* method was to re-adjust the particle’s velocity such that particles just lands on the closest boundary along its path. Figure 3 shows an initial feasible location X_{not} which gets updated to an infeasible X_C. To make X_C feasible the solution is dragged back along its line of movement till it reaches the nearest boundary. It should be noted that this mechanism belongs to group

(B), as the movement is carried out vectorically rather than along one particular dimension. In the original study, this method was useful in solving problems with optima close to the boundary. However, it should be noted that authors had employed a turbulence factor (t_f) to promote diversity in the swarm (which we have completely exempted in this study and hence a poor performance of SHR. method will be found).

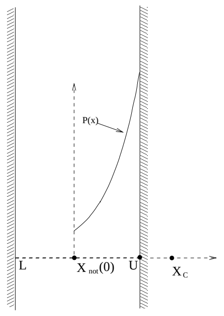

Fig. 4. Variable wise *Exponential Distribution* strategy for handling bounds.

Exponential Distribution: This method is similar to *EXP.*, which was proposed in [1]. According to this approach a particle is brought back inside the search space, dimension-wise, in the region between its old position and the bound. The new particle positions are sampled in such a manner that higher probability is assigned to regions near the boundary, and the probability of sampling a location decreases exponentially as one moves away from the boundary. Consider Figure 4, in which particle was located at X_{not}, along some particular dimension, and on position update it reaches X_C, where it violates the upper-bound (U). Now, in order to make X_C feasible following probability distribution function is considered:

$$P(x) = Ae^x \ s.t. \ \ 0 \leq x \leq U - X_{not} \tag{2}$$

In above equation, A is some constant which is found by computing and equating cumulative probability to 1:

$$\int_0^{U-Xnot} Ae^x \ dx = 1 \tag{3}$$

$$\implies A = \frac{1}{e^{(U-Xnot)} - 1} \tag{4}$$

Let r be a uniformly distributed random number in [0, 1], then x' is sampled between X_{not} and U according to above distribution as follows:

$$r = \int_0^d Ae^x \; dx \tag{5}$$
$$\implies d = \log(1 + r(e^{U - X_{not}} - 1)) \tag{6}$$
$$\implies x' = X_{not} + d \tag{7}$$

The above calculations are valid for the case when X_C is greater than U. In situations, when the updated location X_C is lesser than L (figure omitted), then following probability density function is considered:

$$P(x) = Ae^x \; s.t. \;\; 0 \leq x \leq X_{not} - L \tag{8}$$

and x' is computed along similar lines:

$$\int_0^{Xnot-L} Ae^x \; dx = 1 \tag{9}$$
$$\implies A = \frac{1}{e^{(Xnot-L)} - 1} \tag{10}$$
$$r = \int_0^d Ae^x \; dx \tag{11}$$
$$\implies d = \log(1 + r(e^{X_{not} - L} - 1)) \tag{12}$$
$$\implies x' = X_{not} - d \tag{13}$$

It should be noted that according to this method particles are re-sampled in the region between X_{not} and U(or L), irrespective of the distance between its location X_C and violated boundary (U or L). The method of *Exponential Distribution* in general is applicable with any evolutionary method.

2.2 Proposed Bound Handling Methods

The exponential probability distribution function described in previous section brings particles back into the search space variable wise and ignores the particle distance from the violated boundary. The distance by which particle exceeds the boundary can also provide useful information. One way to utilize this distance information is to bring particles back into the search space with higher probabilities at the boundary when *falling-out* distance is small. In situations when particles are too far outside the search space, i.e. the *falling-out* distance is large, particles be brought back more uniformly. Since the distribution of particles back into search space varies, it is unlikely that search becomes stagnated.
Consider a scenario in which particle was originally located at a vector point X_{not} and after updation it moves to a new vector location X_C (which is infeasible). The goal is to bring particle back into the feasible region along the vector joining X_{not} and X_C. For the purpose of illustration, we consider a case where X_C violates the bound (U) along some particular dimension as shown in Figure 5. It should be noted that, in general, more than one bounds may be violated. In such case,

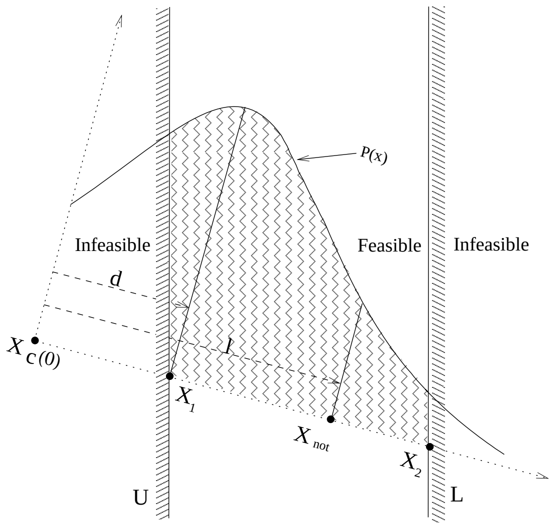

Fig. 5. Vector based *Adaptive Distribution* strategy for handling bounds.

the bound intersecting the line joining X_{not} and X_C and lying closest to the old location (X_{not}) is selected. Let the intersection of selected bound and line joining X_{not} and X_C be X_1.
Let the intersection of the line joining X_C and X_{not} with bound on the opposite side be X_2. At this stage we propose two strategies, namely *Adaptive Spread Distribution* and *Adaptive Confined Distribution*, to re-sample a location x' in region between X_1 and X_2. Both, these strategies utilize the following probability distribution function:

$$P(x) = \frac{a}{(x-d)^2 + \alpha^2 d^2} \; s.t. \;\; 0 \leq x \leq \infty \tag{14}$$

In above equation a is a constant to be determined and α is kept as a user defined parameter (we choose α equal to 1.2 in this study). According to the Figure 5, it can be seen that the proposed probability density function has a peak at location X_1. The peak is heightened if α is lowered. The calculation for the distribution constant a is done by equating the cumulative probability equal to one. The limits are chosen from X_C (taken as origin) till infinity.

$$\int_0^\infty \frac{a}{(x-d)^2 + \alpha^2 d^2} \; dx = 1 \tag{15}$$

$$\implies \quad \frac{a}{\alpha^2 d^2} (\tan^{-1} \frac{x-d}{\alpha^2 d^2})_0^\infty = 1 \tag{16}$$

$$\implies \quad a = \frac{\alpha^2 d^2}{\frac{\pi}{2} + \tan^{-1} \frac{1}{\alpha}} \tag{17}$$

1. Adaptive Spread Distribution: This strategy aims to sample a location between (and inclusive of) X_1 and X_2, thereby maintaining diversity while bringing the particles back into the feasible region. The bringing back is done by redistributing the probability in infeasible region probability into the feasible region as follows:

$$let,\int_{d}^{|X_2-X_1|}\frac{a}{(x-d)^2+\alpha^2d^2}\;dx=p_1 \tag{18}$$

$$\tag{19}$$

Then, the probability distribution function is reconstructed as:

$$P_1(x)=\frac{a}{p_1((x-d)^2+\alpha^2d^2)}\;s.t.\;\; d\leq x\leq |X_2-X_1| \tag{20}$$

Let X' denote the sampled location, r be a uniformly distributed random number in [0,1] then $|X'|$ can be found as follows:

$$r\;=\int_{d}^{|X'|}\frac{a}{p_1((x-d)^2+\alpha^2d^2)}\;dx \tag{21}$$

$$\implies |X'|=d+\alpha d\tan(r\tan^{-1}\tfrac{(|X_2|-d)}{\alpha d}) \tag{22}$$

Once $|X'|$ is calculated, the new vector position X' between X_1 and X_2 can be easily found.

2. Adaptive Confined Distribution: This is similar to method 1, with the only difference that the re-sampled location in this case (denoted as X'') lies between (and inclusive of) X_1 and X_{not}. As the name suggests, bringing back of particle by this method is confined in the region on the line joining old position and the nearest bound. The probability distribution function and computation of a remain same as before. The redistribution of probability is carried out in the region between X_1 and X_{not}, and new location X'' is calculated as follows :

$$let,\int_{d}^{|X_{not}-X_1|}\frac{a}{(x-d)^2+\alpha^2d^2}\;dx=p_2 \tag{23}$$

$$\tag{24}$$

$$P_2(x)=\frac{a}{p_2((x-d)^2+\alpha^2d^2)}\;s.t.\;\; d\leq x\leq |X_{not}-X_1| \tag{25}$$

$$r\;=\int_{d}^{|X''|}\frac{a}{p_2((x-d)^2+\alpha^2d^2)}\;dx \tag{26}$$

$$\implies |X''|=d+\alpha d\tan(r\tan^{-1}\tfrac{(|X_{not}|-d)}{\alpha d}) \tag{27}$$

3 Simulation Scenarios

In general, performance of any bound handling mechanism may depend on the nature of optimization problem, location of the optima, number of variables, bounds on the variables, their relative ranges, and choice of the evolutionary optimizer itself. The most preferred bound handling mechanism would be one which shows consistent and good performance over several scenarios. Single objective optimization problems can either be unimodal or multi-modal. In this study we choose only unimodal problems. Solving multi-modal problem in-itself is a challenging task and an independent line of research.
The goal in this study is to consider couple of standard test problems Ellipsoidal function (F_{elp}) and Schwefel's function (F_{sch}), with 20 variables in three different scenarios, such that for each problem optimum lies (i) on the boundary, (ii) at the center and (iii) just inside the boundary, of the search space.
The problem description of Ellipsoidal function (F_{elp}) and Schwefel's function (F_{sch}) is provided as follows:

$$F_{\text{elp}} = \sum_{i=1}^{n} i x_i^2, \tag{28}$$

$$F_{\text{sch}} = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 \tag{29}$$

In unconstrained search space, the two problems have their minimum at $x_i^* = 0$ with $F^* = 0$. F_{elp} is a separable problem, whereas F_{sch} is non-separable. To invoke different location settings for the optima following variable domains are considered:

(i) $x_i \in [0, 10]$
(ii) $x_i \in [-10, 10]$
(iii) $x_i \in [-0.05, 10]$

As stated before, for (i) optima lies on the search boundary, (ii) optima lies in the center of the search, and (iii) optima lies just inside the boundary (different from being exactly on the boundary). For each test scenario the population initialization is done uniformly in the search domain. Although, it is suggested somewhere else [4, 3] that initialization of solutions done uniformly in the search space (or in a region close to optima) does not test an algorithm's ability of progressing towards the optimal region, but we argue that such initialization is acceptable in this case. The focus of this study is not to test an algorithm's ability to find optima efficiently rather study the effects of different bound handling mechanisms.
The results are presented as follows: After initializing the population randomly and uniformly in the search domain, we count the number of function evaluations needed for the algorithm to find a solution close to the optimal solution and we call this our evaluation criterion S_1. We choose a function value of 10^{-10} for this purpose. Choosing a high accuracy of 10^{-10} in the function value as termination criteria minimizes the chances of locating the problem optima due to any random effect, and thereby provides clear insight into the behavior of bound handling mechanism.

To eliminate the random effects and gather results of statistical importance, each algorithm is tested on a test problem 50 times (each run starting with a different initial population). A particular run is concluded if the evaluation criterion S_1 is met, or the number of function evaluations exceed one million. If only a few out of 50 runs are successful then we report the count of successful runs in the bracket. In this case, the best, median and worst number of function evaluations are computed from the successful runs. If none of the runs are successful, we denote this by marking with *(DNC)* (Did Not Converge). In such cases, we report the best, median and worst attained function values of the best solution at the end of each run. Understandably, each of 50 runs is run for a maximum of one million function evaluations. To distinguish the unsuccessful results from successful ones, we present the fitness value information of the unsuccessful runs in italics.

4 Results and Discussions

Once the particle is brought back into the search space from an infeasible location, what happens to its velocity is a matter of importance. One approach could be to modify the particle velocity based on new feasible location (i.e. $V_t + 1 = X'_{t+1} - X_t$, where X_t is the original feasible location, X_{t+1} is the infeasible particle position obtained on updation, and X'_{t+1} is new made feasible location for X_{t+1}). This case is indicated as 'Vel. Recomputed'. The second approach could be to keep the original velocity based on which infeasible location was found (i.e. $X_{t+1} = V_{t+1} + X_t$). For *Adaptive Spread Distribution*, *Adaptive Confined Distribution* and *Exponential Distribution* only first approach for velocity computation is applied. For *Periodic*, *Random* and *SetOnBoundary* both the approaches are tested. For *SetOnBoundary* additional strategy of velocity reflection is also tested. Velocity reflection simply means that if a particle is set on the i^{th} boundary, then v_i^{t+1} is changed to $-v_i^{t+1}$. The goal of the velocity reflection is to explicitly allow particles to move into the search space. For *SHR*. the particle is placed on the boundary as discussed earlier and velocity is set to zero. Thus, a total of 11 different bound handling cases are tested. The simulations are performed for both the test problems, with three different cases of optimal location, and results are presented in Tables 1 to 6. Following conclusions can be drawn based on tabulated results. *Adaptive Spread Distribution* and *Exponential Distribution* never fail. The *SHR*. method always shows *(DNC)*. The *Periodic* is only successful with the both test problems when optimum lies at the center. The performance of *Periodic* with re-computed velocity is always better compared to keeping the velocity unchanged. The *Random* technique also shows convergence when optimum lies at the center. Compared to keeping velocity unchanged, re-computing the velocity shows better performance with *Random*. In situations where convergence is obtained, *Random* approach is better or at-least similar, as compared to *Periodic*. In cases of (DNC), *Random* is still better than *Periodic* in terms of best fitness value achieved.
For both the test functions when optimum is at the boundary i.e. Tables 4 and 1, performances of *SetOnBoundary* with *Vel. Recomputed* and *SetOnBoundary* with *Vel. Set Zero* can be regarded best for F_{elp},F_{sch}, respectively. However, all 50 runs are not successful. The *SetOnBoundary* boundary schemes are always expected to work well for problems with optimum one the boundary. The *Exponential Distribution* performs second best and *Adaptive Spread Dist.* third best, for the test

Table 1. F_{elp} with optimum on the boundary

Strategy	**Best**	**Median**	**Worst**
Adap. Spread Dist.	36,300	47,500	60,800
Adap. Confined Dist.	50,900(49)	86,000	159,000
Exponential Dist.	4,600	5,900	9,100
Periodic(Vel. Recomputed)	*3.94e+02 (DNC)*	*6.63e+02 (DNC)*	*1.17e+03 (DNC)*
Periodic(Vel. Unchanged)	*8.90e+02 (DNC)*	*1.03e+02 (DNC)*	*1.34e+03 (DNC)*
Random(Vel. Recomputed)	*1.95e+01 (DNC)*	*3.48e+01 (DNC)*	*7.42e+01 (DNC)*
Random(Vel. Unchanged)	*4.70e+02 (DNC)*	*6.81e+02 (DNC)*	*11.31e+02 (DNC)*
SetOnBoundary(Vel. Recomputed)	900 (44)	1,300	5,100
SetOnBoundary(Vel. Reflected)	242,100	387,100	811,400
SetOnBoundary(Vel. Set Zero)	1,300	1,900	4,100
SHR.	*11.91e+02 (DNC)*	*18.96e+02 (DNC)*	*32.99e+02 (DNC)*

Table 2. F_{elp} with optimum in the center

Strategy	**Best**	**Median**	**Worst**
Adap. Spread Dist.	30,900	34,200	39,500
Adap. Confined Dist.	31,200	33,600	38,700
Exponential Dist.	31,800	34,900	40,500
Periodic(Vel. Recomputed)	32,200	35,100	37,900
Periodic(Vel. Unchanged)	33,800	36,600	41,200
Random(Vel. Recomputed)	32,200	34,600	36,600
Random(Vel. Unchanged)	31,000	34,900	37,500
SetOnBoundary(Vel. Recomputed)	31,900	35,500	40,500
SetOnBoundary(Vel. Reflected)	50,800(38)	83,200	484,100
SetOnBoundary(Vel. Set Zero)	31,600	35,000	37,200
SHR.	*1.04e+02 (DNC)*	*4.02e+02 (DNC)*	*9.87e+02 (DNC)*

functions. *Adaptive Confined Dist.* shows convergence 49 out of 50 times for F_{elp} and *(DNC)* for F_{sch}. It is important to mention that with *Adaptive Distributions* lowering α resulted in better performance in case of boundary optimum. For the sake of brevity we exclude those results.

Tables 2 and 5 indicate that when optimum is at the center there is only a small variation in number of function evaluations for different bound handling techniques. This can be attributed to the fact that tendency of the particles in PSO to

Table 3. F_{elp} with optimum close to the edge of boundary

Strategy	**Best**	**Median**	**Worst**
Adap. Spread Dist.	28,700	33,400	39,300
Adap. Confined Dist.	32,200 (49)	48,300	80,700
Exponential Dist.	23,100	28,000	80,700
Periodic(Vel. Recomputed)	*4.26e+02 (DNC)*	*5.87e+02 (DNC)*	*8.98e+02 (DNC)*
Periodic(Vel. Unchanged)	*6.46e+02 (DNC)*	*10.06e+02 (DNC)*	*12.38e+02 (DNC)*
Random(Vel. Recomputed)	*1.33e+01 (DNC)*	*2.82e+01 (DNC)*	*5.93e+01 (DNC)*
Random(Vel. Unchanged)	*4.59e+02 (DNC)*	*6.59e+02 (DNC)*	*9.37e+02 (DNC)*
SetOnBoundary(Vel. Recomputed)	*2.5e-01 (DNC)*	*3.9e-01 (DNC)*	*50.03e+01 (DNC)*
SetOnBoundary(Vel. Reflected)	388,700 (39)	757,200	977,400
SetOnBoundary(Vel. Set Zero)	*2.1e-01 (DNC)*	*3.2e-01 (DNC)*	*2.0e+02 (DNC)*
SHR.	*11.69e+02 (DNC)*	*18.58e+02 (DNC)*	*32.61e+02 (DNC)*

fly out of the search space is diminished when optimum as located in middle of the search space. Thus, the effect of boundary handling mechanism is not significant. Here, *Adap. Spread Dist.* performs best with F_{elp} and *Adap. Confined Dist.* performs second best with F_{sch}.

The third scenario, where the optimum is located close to the boundary is interesting, Tables 3 and 6. *Exponential Dist.* performs best in terms of number of number of function evaluations. However, shows success 47 times for F_{sch}. *Adap. Spread Dist.* performs second best and is successful in all 50 runs for both the test functions. The *Adap. Confined Dist.* method convergence but takes larger number of function evaluations and is not successful in all 50 runs. The only other method successful in this scenario is *SetOnBoundary(Vel. Reflected)* but number of function evaluations required are quite large.

Overall, it can concluded that *Exponential Distribution* and *Adaptive Spread Distribution* are found to be most robust and consistently perform well, with former comparatively doing better. A key difference in these two distributions is the variation of probability distributions depending on particle's position. In *Exponential Distribution* the importance is given to the distance between the original particle location (which is feasible) and the bound it crosses. Larger this distance, more uniformly the particle is brought inside i.e. probability of sampling a location close to boundary is relatively smaller. If the particle is close to boundary and falls out then it has higher probability of being placed closer to the boundary. Since each bound is handled separately it can be argued that this distribution is more explorative (even though the particles are brought back-in between original location and bound only).

Table 4. F_{sch} with optimum on the boundary

Strategy	**Best**	**Median**	**Worst**
Adap. Spread Dist.	95,000 (47)	213,900	769,100
Adap. Confined Dist.	*1.20e-04 (DNC)*	*1.75e+01 (DNC)*	*7.74e+02 (DNC)*
Exponential Dist.	5,000	6,000	12,600
Periodic(Vel. Recomputed)	*4.85e+03 (DNC)*	*7.82e+03 (DNC)*	*1.34e+04 (DNC)*
Periodic(Vel. Unchanged)	*7.69e+03 (DNC)*	*1.11e+04 (DNC)*	*1.51e+04 (DNC)*
Random(Vel. Recomputed)	*2.75e+02 (DNC)*	*5.59e+02 (DNC)*	*1.01e+03 (DNC)*
Random(Vel. Unchanged)	*5.26e+03 (DNC)*	*7.84e+03 (DNC)*	*1.03e+04 (DNC)*
SetOnBoundary(Vel. Recomputed)	*8.0e+02 (DNC)*	*1.10e+03 (DNC)*	*3.90e+03 (DNC)*
SetOnBoundary(Vel. Reflected)	172,000	242,000	434,000
SetOnBoundary(Vel. Set Zero)	1,000 (40)	1,600	5,300
SHR.	*1.43e+04 (DNC)*	*2.14e+04 (DNC)*	*3.52e+04 (DNC)*

Table 5. F_{sch} with optimum in the center

Strategy	**Best**	**Median**	**Worst**
Adap. Spread Dist.	114,300	127,600	147,300
Adap. Confined Dist.	111,700	127,100	149,800
Exponential Dist.	112,600	131,400	148,300
Periodic(Vel. Recomputed)	113,400	130,900	150,600
Periodic(Vel. Unchanged)	121,200	137,800	159,100
Random(Vel. Recomputed)	106,300	128,900	145,700
Random(Vel. Unchanged)	117,600	133,000	148,400
SetOnBoundary(Vel. Recomputed)	118,500 (49)	132,300	161,100
SetOnBoundary(Vel. Reflected)	*3.30e-06 (DNC)*	*8.32e+01 (DNC)*	*2.95e+02 (DNC)*
SetOnBoundary(Vel. Set Zero)	112,000	132,000	149,000
SHR.	*5.24e+01 (DNC)*	*1.33e+02 (DNC)*	*4.80e+02 (DNC)*

On the other hand *Adaptive Distributions* take into account the distance of the particle out side the search region from a variable bound. The particle is brought back-in more uniformly within or closer to the boundary depending on whether this distance is larger or smaller, respectively. According to this method, the par-

Table 6. F_{sch} with optimum close to the edge of boundary

Strategy	**Best**	**Median**	**Worst**
Adap. Spread Dist.	107,500	150,700	554,600
Adap. Confined Dist.	152,600 (11)	323,900	447,500
Exponential Dist.	77,300(47)	103,100	259,800
Periodic(Vel. Recomputed)	*4.54e+03 (DNC)*	*7.63e+03 (DNC)*	*1.29e+04 (DNC)*
Periodic(Vel. Unchanged)	*7.33e+03 (DNC)*	*1.06e+04 (DNC)*	*1.46e+04 (DNC)*
Random(Vel. Recomputed)	*1.88e+02 (DNC)*	*4.02e+02 (DNC)*	*9.48e+02 (DNC)*
Random(Vel. Unchanged)	*4.66e+03 (DNC)*	*7.25e+03 (DNC)*	*8.98e+03 (DNC)*
SetOnBoundary(Vel. Recomputed)	*3.0e-02 (DNC)*	*4.39e-01 (DNC)*	*2.16e+03 (DNC)*
SetOnBoundary(Vel. Reflected)	361,900	514,500	720,200
SetOnBoundary(Vel. Set Zero)	*2.0e-02 (DNC)*	*1.48e-01 (DNC)*	*1.52e+03 (DNC)*
SHR.	*1.38e+04 (DNC)*	*2.09e+04 (DNC)*	*3.46e+04 (DNC)*

ticles always lie on the vector joining original feasible and updated infeasible location. Compared to the *Exponential Distribution*, this method is more restrictive or less explorative, and the variables are modified inter-dependently. Out of the two versions of adaptive distribution methods, spread is obviously more exploratory along the line compared to the confined. Drawing general inferences as to which of these methods is better is difficult at this stage.

SetOnBoundary(Vel. Reflected) showed convergence on several cases. This could be explained based on the fact that negating the velocity may be better than (1) setting it to zero (which would imply that exploration for a particle is terminated) and (2) re-computing velocity(which would only reduce the effect of velocity and eventually stop exploration in next step when particle again flies out of the search space). By reflection the search is re-initialized in same region but now in opposite direction and this method promotes particles to search the feasible spaces in next steps.

Periodic and *Random* methods do not utilize information from particle locations and a rewarding performance is unexpected. Failure of *SHR.* method may be accounted to the fact that particle velocity was set to zero and no turbulence factor was used.

5 Conclusion

The paper compared existing and two newly proposed *bound handling* methods empirically in context to PSO under varying simulation scenarios. It was found that performance of the optimizer dependent on the *bound handling* strategy employed and location of the optimum. It was found that *Exponential Distribution* and *Adaptive Spread Distribution* methods are most robust and never failed in solving the optimization problems. The other strategies for *bound handling* were too deterministic and possibly functioned without any useful information from particle locations. From the simulation results it can be concluded that probabilistic way of bringing the solutions back-into the search space is an effective strategy and guarantees accurate performance which is independent of optimum location. In future, application of *Exponential Distribution* and *Adaptive Distributions* can be tested with other evolutionary approaches, based on their general form of applicability.

References

1. Alvarez-Benitez, J.E., Everson, R.M., Fieldsend, J.E.: A MOPSO algorithm based exclusively on pareto dominance concepts. In: EMO. vol. 3410, pp. 459–473 (2005)
2. Deb, K.: An efficient constraint handling method for genetic algorithms. In: Computer Methods in Applied Mechanics and Engineering. pp. 311–338 (1998)
3. Deb, K., Annand, A., Jhoshi, D.: A computationally efficient evolutionary algorithm for real-parameter optimization. Evol. Comput. 10(4), 371–395 (2002)
4. Deb, K., Padhye, N.: Development of efficient particle swarm optimizers by using concepts from evolutionary algorithms. In: Proceedings of the 12th annual conference on Genetic and evolutionary computation. pp. 55–62 (2010)
5. Helwig, S., Wanka, R.: Particle swarm optimizatio in high-dimensional bounded search spaces. In: Proceedings of the 2007 IEEE Swarm Intelligence Symposium. pp. 198–205
6. Padhye, N., Branke, J., Mostaghim, S.: Empirical comparison of mopso methods - guide selection and diversity preservation -. In: Proceedings of CEC. pp. 2516 – 2523. IEEE (2009)
7. Reklaitis, G.V., Ravindran, A., Ragsdell, K.M.: Engineering Optimization Methods and Applications. Willey, New York (1983)
8. Zhang, W.J., Xie, X.F., Bi, D.C.: Handling boundary constraints for numericaloptimization by particle swarm flying in periodic search space. In: Proceedings of Congress on Evolutionary Computation. pp. 2307–2311 (2004)