Boundary Handling Approaches in Particle Swarm Optimization

Nikhil Padhye
Department of Mechanical Engineering
Massachusetts Institute of Technology, Cambridge, MA-02139
npdhye@mit.edu

Kalyanmoy Deb
Department of Mechanical Engineering
Indian Institute of Technology Kanpur, Kanpur-208016, U.P., India
deb@iitk.ac.in

Pulkit Mittal
Department of Electrical Engineering
Indian Institute of Technology Kanpur, Kanpur-208016, U.P., India
pulkitm@iitk.ac.in

KanGAL Report Number 2012014

July 28, 2012

Abstract

In recent years, Particle Swarm Optimization (PSO) methods have gained popularity in solving single objective and other optimization tasks. In particular, solving constrained optimization problems using swarm methods has been attempted in past but arguably stays as one of the challenging issues. A commonly encountered situation is one in which constraints manifest themselves in form of variable bounds. In such scenarios the issue of constraint-handling is somewhat simplified. This paper attempts to review popular *bound handling* methods, in context to PSO, and proposes new methods which are found to be robust and consistent in terms of performance over several simulation scenarios. The effectiveness of *bound handling* methods is shown PSO; however the methods are general and can be combined with any other optimization procedure.

1 Introduction

Optimization problems are wide-spread in several domains of science and engineering. The usual goal is to minimize or maximize some pre-defined objective(s) and specified constraints. Without any loss of genrality, 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:

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

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 [10, 2].

In context to Particle Swarm Optimization (PSO), several bound handling methods have already been proposed [8, 6, 1, 7]. However, many of these past proposals exploit the information about location of the optimum and fail to perform when location of optimum changes [9]. The goal of this paper is to come up with robust bound handling techniques which never fail 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 four standard test problems under different scenarios and their performances are compared.

The rest of the paper is organized as follows: Section 2 reviews different bound handling techniques and provides a detailed description two newly proposed adaptive bound handling methods. Section 3 provides a description on the test problems, simulation carried out along with results and discussions. Finally, conclusions are made in Section 4.

2 Bound Handling Mechanisms

When objective function is not defined in the infeasible region only following alternatives are available: (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 [3]. Particularly, in context to PSO we need 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 meth-*

ods. The *bound handling methods* can be broadly divided into two groups *A* and *B*. Group *A* techniques carry out feasibility search variable wise, whereas group *B* techniques 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. In the 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 *Inverse Parabolic (IP) Distribution* is proposed in this paper 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

2.1.1 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. If X_c is the current infeasible variable location with L and U denoting the upper and lower bounds for the corresponding dimension (the same notation is used for rest of the text), then a new feasible location is selected randomly in [L,U].

2.1.2 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 [11]. This is done by placing repeated copies of original search space along the dimension of interest. 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:

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

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.

2.1.3 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:

$$X_C \rightarrow X_C^{new} = \left\{ \begin{array}{ll} L & \text{IF } X_C < \text{L} \\ U & \text{IF } X_C > \text{U} \end{array} \right\}$$

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.

2.1.4 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. 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.

2.1.5 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 particle's old position and the violated 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. We have tried two version for *EXP.* method: (i) the new position is re-sampled between the particle's original location and the bound (lower or upper) that has been violated, and (ii) the new position is re-sampled in the entire region between between the upper and lower bounds of the dimension being violated.

2.2 Proposed Boundary Handling Methods

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, Figure 1, 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

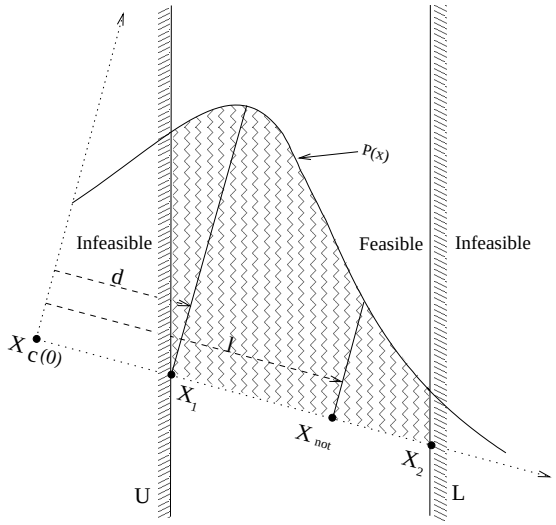

Figure 1: Vector based *Inverse Parabolic Distribution* strategy for handling bounds.

case where X_C violates the bound (U) along some particular dimension as shown in Figure 1. It should be noted that, in general, more than one bounds may be violated. In such case, 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 *Inverse Parabolic Spread Distribution* and *Inverse Parabolic 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:

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

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 1, 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.

$$\int_0^\infty \frac{a}{(x-d)^2 + \alpha^2 d^2} \; dx = 1 \implies \quad a = \frac{\alpha^2 d^2}{\frac{\pi}{2} + \tan^{-1}\frac{1}{\alpha}} \tag{3}$$

1. Inverse Parabolic 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:

$$let, \quad \int_{d}^{|X_2-X_1|+d} \frac{a}{(x-d)^2+\alpha^2 d^2}\ dx = p_1 \tag{4}$$

Then, the probability distribution function is reconstructed as:

$$P_1(x) \quad = \frac{a}{p_1((x-d)^2+\alpha^2 d^2)} \quad s.t.\ \ d \le x \le |X_2-X_1|+d \tag{5}$$

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

$$r \quad = \int_{d}^{|X'|} \frac{a}{p_1((x-d)^2+\alpha^2 d^2)}\ dx \tag{6}$$

$$\implies \quad |X'| = d + \alpha d \tan(r \tan^{-1} \tfrac{(|X_2|-d)}{\alpha d}) \tag{7}$$

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

2. Inverse Parabolic 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 :

$$let, \quad \int_{d}^{|X_{not}-X_1|+d} \frac{a}{(x-d)^2+\alpha^2 d^2}\ dx = p_2 \tag{8}$$

$$P_2(x) \quad = \frac{a}{p_2((x-d)^2+\alpha^2 d^2)} \quad s.t.\ \ d \le x \le |X_{not}-X_1|+d \tag{9}$$

$$r \quad = \int_{d}^{|X''|} \frac{a}{p_2((x-d)^2+\alpha^2 d^2)}\ dx \tag{10}$$

$$\implies \quad |X''| = d + \alpha d \tan(r \tan^{-1} \tfrac{(|X_{not}|-d)}{\alpha d}) \tag{11}$$

3 Simulations and Results

We have considered four standard test problems: Ellipsoidal (F_{elp}), Schwefel (F_{sch}), Ackeley (F_{ack}) and Rosenbrock (F_{ros}) with 20 variables. Three different scenarios are considered for each problem such that optimum lies (i) on the boundary, (ii) in

the center, and (iii) just inside the boundary of the search space. The test problems are given as follows:

$$F_{\text{elp}} = \sum_{i=1}^{n} i x_i^2 \tag{12}$$

$$F_{\text{sch}} = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 \tag{13}$$

$$F_{\text{ack}} = -20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{i=n} x_i^2} \right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i) \right) + 20 + e \tag{14}$$

$$F_{\text{ros}} = \sum_{i=1}^{n-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \tag{15}$$

F_{elp}, F_{sch} and F_{elp} have their minimum at $x_i^* = 0$, whereas F_{ros} has its minimum at $x_i^* = 1$. All the functions have minimum value of $F^* = 0$. F_{elp} is the only variable separable problem.

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. We choose a function value of 10^{-10} for F_{elp}, F_{sch} and F_{elp} and 10^{-3} for F_{ros} (which is a relatively difficult problem). This is similar to what has been proposed in [5, 4].

Each algorithm is tested on a test problem 50 times (each run starting with a different initial population). A particular run is concluded if termination criterion 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.

Once the particle is brought back into the search space the velocity can either be left unchanged or re-computed. By re-computed we mean that if X'_{t+1} is the new feasible location corresponding to X_t, then the velocity is re-adjusted as $V_{t+1} = X'_{t+1} - X_t$. For *Inverse Parabolic Spread Distribution*, *Inverse Parabolic Confined Distribution* and *Exponential Distribution* the velocity is re-computed. For *Periodic*, *Random* and *SetOnBoundary* velocity is either left unchanged and re-computed. For *SetOnBoundary* additional strategy of velocity reflection is also tried i.e. 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 back into the search space. For *SHR.* the particle is placed on the boundary as discussed earlier and velocity is set to zero. To this end, a total of 13 different bound handling cases are tested. The simulations results for four test problems in three different settings are provided in Tables 1 to 4.

Following key inferences can be drawn from the tabulated results:

1. The *bound handling methods* show a large variation in the performance depending on the choice of test problem and location of the optimum with respect to the search space.

2. The performance of *bound handling* is comparable when optimum lies in the center. This can be understood intuitively from the fact that tendency of particles to fly out of the search space is little when the optimum is in the center of the search space. For e.g., the *Periodic* methods fail in all the cases but are able to show convergence for all the test problems when optimum is in the center. When optimum is on the boundary or close to the boundary then effect of bound handling method becomes critical.
3. *Inverse Parabolic Spread Distribution* never failed in any of the 12 simulation scenarios. *Inverse Parabolic Confined Distribution*, *Exponential Confined Distribution* and *Exponential Spread Distribution* are successful in 11, 10 and 8 cases, respectively. It is speculated that proposed IP distributions allow greater chances of creating an offspring close to the boundary then the exponential probability distribution, and hence the performance using IP method is better.
4. *SHR. (Vel. Recomputed and Vel. Set Zero)* methods succeeded in 10 cases. *Set on Boundary: Vel. Recomputed, Vel., Reflected and Vel. SetZero* succeeded 7, 7, 9 times, respectively. Random (Velocity re-computed and Vel. Set Zero) succeed in 5 cases.
5. On lowering α *Inverse Parabolic Distributions* showed an improvement in the performance. For the sake of brevity we exclude those results.

4 Conclusion

In this paper, we have compared existing and newly proposed *bound handling* methods in context to PSO. Four test problems with three different settings of optimum with respect to the search space were tried. The performance of the *bound handling* strategy dependent upon the test problem and location of the optimum. *Inverse Parabolic Spread Distribution* was found to be most the most robust method and never failed. *Inverse Parabolic Confined Distribution* and *Exponential Spread Distribution* were found competitive. *SHR.* methods were successful too but took larger number of function evaluations. Other bound handling strategies were either too deterministic or did not utilize the information of the particle's location properly. It can be concluded that probabilistic way of bringing the solutions back into the search space while utilizing the information of their initial location is an appropriate approach, and guarantees a robust performance. Although, the illustrations made in this paper are in context to PSO, but the bound handling strategies can be applied to any other evolutionary algorithm.

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.: Optimization for Engineering Design: Algorithms and Examples. Prentice-Hall, New Delhi (1995)

[3] Deb, K.: An efficient constraint handling method for genetic algorithms. In: Computer Methods in Applied Mechanics and Engineering. pp. 311–338 (1998)

[4] Deb, K., Annand, A., Jhoshi, D.: A computationally efficient evolutionary algorithm for real-parameter optimization. Evol. Comput. 10(4), 371–395 (2002)

[5] 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)

[6] 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

[7] Helwig, S., Branke, J., Member, S.M.: Experimental analysis of bound handling techniques in particle swarm optimization. IEEE Transactions on Evolutionary Computation (99) (2012)

[8] 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)

[9] Padhye, N.: Development of Efficient Particle Swarm Optimizers and Bound Handling Methods. Master's thesis, IIT Kanpur, India (2010)

[10] Reklaitis, G.V., Ravindran, A., Ragsdell, K.M.: Engineering Optimization Methods and Applications. Willey, New York (1983)

[11] 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)

Table 1: Results on F_{elp}

Strategy	**Best**	**Median**	**Worst**
F_{elp} in [0,10]			
IP Spread Dist.	39,900	47,000	67,000
IP Confined Dist.	47,900 (49)	88,600	140,800
Exponential Spread Dist.	*3.25e-01*	*5.02e-01*	*1.08e+00*
Exponential Confined Dist.	4,600	5,900	7,500
Periodic(Vel. Recomputed)	*3.94e+02 (DNC)*	*6.63e+02 (DNC)*	*1.17e+03(DNC)*
Periodic(Vel. Unchanged)	*8.91e+02 (DNC)*	*1.03e+03 (DNC)*	*1.34e+03 (DNC)*
Random(Vel. Recomputed)	*1.97e+01 (DNC)*	*3.37e+01 (DNC)*	*8.10e+01(DNC)*
Random(Vel. Unchanged)	*5.48e+02 (DNC)*	*6.69e+02 (DNC)*	*9.65e+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 (48)	1,900	4,100
SHR.(Vel. Recomputed)	8,200 (49)	10,900	14,300
SHR.(Vel. Set Zero)	33,000	40,700	53,900
F_{elp} in [-10,10]			
IP Spread Dist.	31,600	34,000	37,900
IP Confined Dist.	30,900	33,800	38,500
Exponential Spread Dist.	30,500	34,700	38,300
Exponential Confined Dist.	31,900	35,100	38,200
Periodic(Vel. Recomputed)	32,200	35,100	37,900
Periodic(Vel. Unchanged)	33,800	36,600	41,200
Random(Vel. Recomputed)	31,900	34,800	37,400
Random(Vel. Unchanged)	31,600	34,900	38,100
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.(Vel. Recomputed)	32,000	34,400	48,200
SHR.(Vel. Set Zero)	31,400	34,000	37,700
F_{elp} in [-1,10]			
IP Spread Dist.	28,200	31,900	35,300
IP Confined Dist.	28,300	32,900	44,600
Exponential Spread Dist.	28,300	30,700	33,200
Exponential Confined Dist.	29,500	33,000	44,700
Periodic(Vel. Recomputed)	*4.86e+01 (DNC)*	*1.41e+02 (DNC)*	*4.28e+02 (DNC)*
Periodic(Vel. Unchanged)	*2.84e+02 (DNC)*	*5.46e+02 (DNC)*	*8.28e+02 (DNC)*
Random(Vel. Recomputed)	36,900	41,900	45,600
Random(Vel. Unchanged)	*1.13e+02 (DNC)*	*2.26e+02 (DNC)*	*4.35e+02 (DNC)*
SetOnBoundary(Vel. Recomputed)	*1.80e+01 (DNC)*	*7.60e+01 (DNC)*	*3.00e+02 (DNC)*
SetOnBoundary(Vel. Reflected)	*2.13e-01 (DNC)*	*2.17e+01 (DNC)*	*1.06e+02 (DNC)*
SetOnBoundary(Vel. Set Zero)	31,700 (2)	31,700	32,600
SHR.(Vel. Recomputed)	29,500 (6)	36,100	42,300
SHR.(Vel. Set Zero)	28,400 (36)	32,700	65,600

Table 2: Results on F_{sch}

Strategy	**Best**	**Median**	**Worst**
F_{sch} in [0,10]			
IP Spread Dist.	67,200	257,800	970,400
IP Confined Dist.	112,400 (6)	126,500	145,900
Exponential Spread Dist.	*3.79e+00 (DNC)*	*8.37e+00 (DNC)*	*1.49e+01 (DNC)*
Exponential Confined Dist.	4,900	6,100	13,500
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)	*2.61e+02 (DNC)*	*5.44e+02 (DNC)*	*1.05e+03 (DNC)*
Random(Vel. Unchanged)	*5.30e+03 (DNC)*	*7.60e+03 (DNC)*	*1.22e+04 (DNC)*
SetOnBoundary(Vel. Recomputed)	800 (30)	1,100	3,900
SetOnBoundary(Vel. Reflected)	171,500	241,700	434,200
SetOnBoundary(Vel. Set Zero)	1,000 (40)	1,600	5,300
SHR.(Vel. Recomputed)	6,900	9,100	11,600
SHR.(Vel. Set Zero)	17,900	31,900	49,800
F_{sch} in [-10,10]			
IP Spread Dist.	106,700	127,500	144,300
IP Confined Dist.	111,500	130,100	149,900
Exponential Spread Dist.	112,300	131,400	149,000
Exponential Confined Dist.	116,400	131,300	148,200
Periodic(Vel. Recomputed)	113,400	130,900	150,600
Periodic(Vel. Unchanged)	121,200	137,800	159,100
Random(Vel. Recomputed)	112,900	129,800	151,100
Random(Vel. Unchanged)	117,000	130,600	148,100
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)	111,900	132,200	149,700
SHR.(Vel. Recomputed)	111,800 (49)	131,800	183,500
SHR.(Vel. Set Zero)	108,400	125,100	143,600
F_{sch} in [-1,10]			
IP Spread Dist.	107,200	130,400	272,400
IP Confined Dist.	120,100 (44)	171,200	301,200
Exponential Spread Dist.	92,800	109,200	126,400
Exponential Confined Dist.	110,200	127,400	256,100
Periodic(Vel. Recomputed)	*8.09e+02 (DNC)*	*2.01e+03 (DNC)*	*5.53e+03(DNC)*
Periodic(Vel. Unchanged)	*2.16e+03 (DNC)*	*4.36e+03 (DNC)*	*6.87e+03 (DNC)*
Random(Vel. Recomputed)	123,300	165,600	280,000
Random(Vel. Unchanged)	*8.17e+02 (DNC)*	*1.96e+03 (DNC)*	*2.68e+03 (DNC)*
SetOnBoundary(Vel. Recomputed)	*2.50e+00 (DNC)*	*1.25e+01 (DNC)*	*5.75e+02 (DNC)*
SetOnBoundary(Vel. Reflected)	*1.86e+00 (DNC)*	*7.76e+00 (DNC)*	*5.18e+01 (DNC)*
SetOnBoundary(Vel. Set Zero)	*1.00e+00 (DNC)*	*5.00e+00 (DNC)*	*4.21e+02 (DNC)*
SHR.(Vel. Recomputed)	*5.00e-01 (DNC)*	*3.00e+00 (DNC)*	*1.60e+01 (DNC)*
SHR.(Vel. Set Zero)	108,300 (8)	130,300	143,000

Table 3: Results on F_{ack}

Strategy	**Best**	**Median**	**Worst**
F_{ack} in [0,10]			
IP Spread Dist.	150,600 (49)	220,900	328,000
IP Confined Dist.	*4.17e+00 (DNC)*	*6.53e+00 (DNC)*	*8.79e+00 (DNC)*
Exponential Spread Dist.	*2.76e-01 (DNC)*	*9.62e-01 (DNC)*	*2.50e+00 (DNC)*
Exponential Confined Dist.	7,800	9,600	11,100
Periodic(Vel. Recomputed)	*6.17e+00 (DNC)*	*6.89e+00 (DNC)*	*9.22e+00 (DNC)*
Periodic(Vel. Unchanged)	*8.23e+00 (DNC)*	*9.10e+00 (DNC)*	*9.68e+00 (DNC)*
Random(Vel. Recomputed)	*3.29e+00 (DNC)*	*3.40e+00 (DNC)*	*4.19e+00 (DNC)*
Random(Vel. Unchanged)	*6.70e+00 (DNC)*	*7.46e+00 (DNC)*	*8.57e+00(DNC)*
SetOnBoundary(Vel. Recomputed)	800	1,100	2,100
SetOnBoundary(Vel. Reflected)	420,600	598,600	917,400
SetOnBoundary(Vel. Set Zero)	1,100	1,800	3,100
SHR.(Vel. Recomputed)	33,800 (5)	263,100	690,400
SHR.(Vel. Set Zero)	*3.65e+00 (DNC)*	*6.28e+00 (DNC)*	*8.35e+00 (DNC)*
F_{ack} in [-10,10]			
IP Spread Dist.	53,900 (46)	58,600	66,500
IP Confined Dist.	54,800 (49)	59,200	64,700
Exponential Spread Dist.	55,100	59,300	63,600
Exponential Confined Dist.	56,800	59,600	65,000
Periodic(Vel. Recomputed)	55,700 (48)	59,900	64,700
Periodic(Vel. Unchanged)	57,900 (49)	62,100	66,700
Random(Vel. Recomputed)	55,100 (47)	59,400	65,100
Random(Vel. Unchanged)	56,300	59,700	65,500
SetOnBoundary(Vel. Recomputed)	55,100 (49)	58,900	65,400
SetOnBoundary(Vel. Reflected)	86,900 (4)	136,400	927,600
SetOnBoundary(Vel. Set Zero)	53,900 (49)	59,600	67,700
SHR.(Vel. Recomputed)	55,800 (47)	58,700	65,800
SHR.(Vel. Set Zero)	55,700 (49)	58,900	62,000
F_{ack} in [-1,10]			
IP Spread Dist.	54,600 (5)	55,100	56,600
IP Confined Dist.	63,200 (1)	63,200	63,200
Exponential Spread Dist.	51,300	55,200	58,600
Exponential Confined Dist.	*1.42e+00(DNC)*	*2.17e+00 (DNC)*	*2.92e+00 (DNC)*
Periodic(Vel. Recomputed)	*2.88e+00 (DNC)*	*4.03e+00 (DNC)*	*5.40e+00 (DNC)*
Periodic(Vel. Unchanged)	*6.61e+00 (DNC)*	*7.46e+00 (DNC)*	*8.37e+00 (DNC)*
Random(Vel. Recomputed)	60,300 (45)	66,200	72,200
Random(Vel. Unchanged)	*4.21e+00 (DNC)*	*4.93e+00 (DNC)*	*6.11e+00 (DNC)*
SetOnBoundary(Vel. Recomputed)	*2.74e+00 (DNC)*	*3.16e+00 (DNC)*	*3.36e+00 (DNC)*
SetOnBoundary(Vel. Reflected)	824,700 (1)	824,700	824,700
SetOnBoundary(Vel. Set Zero)	*1.70e+00 (DNC)*	*2.63e+00 (DNC)*	*3.26e+00 (DNC)*
SHR.(Vel. Recomputed)	*1.45e+00 (DNC)*	*2.34e+00 (DNC)*	*2.73e+00 (DNC)*
SHR.(Vel. Set Zero)	*2.01e+00 (DNC)*	*3.96e+00 (DNC)*	*6.76e+00 (DNC)*

Table 4: Results on F_{ros}

Strategy	**Best**	**Median**	**Worst**
F_{ros} in [1,10]			
IP Spread Dist.	89,800	195,900	243,300
IP Confined Dist.	23,800	164,300	209,300
Exponential Spread Dist.	*9.55e-01 (DNC)*	*2.58e+00 (DNC)*	*7.64e+00 (DNC)*
Exponential Confined Dist.	3,700	128,100	344,400
Periodic(Vel. Recomputed)	*1.24e+04 (DNC)*	*2.35e+04 (DNC)*	*4.24e+04 (DNC)*
Periodic(Vel. Unchanged)	*6.99e+04 (DNC)*	*1.01e+05 (DNC)*	*1.45e+05 (DNC)*
Random(Vel. Recomputed)	*6.00e+01 (DNC)*	*1.37e+02 (DNC)*	*4.42e+02 (DNC)*
Random(Vel. Unchanged)	*2.32e+04 (DNC)*	*3.90e+04 (DNC)*	*8.22e+04 (DNC)*
SetOnBoundary(Vel. Recomputed)	900(45)	1,600	89,800
SetOnBoundary(Vel. Reflected)	*2.14e-03 (DNC)*	*6.01e+02 (DNC)*	*5.10e+04 (DNC)*
SetOnBoundary(Vel. Set Zero)	1,400(48)	3,000	303,700
SHR.(Vel. Recomputed)	3,900(44)	5,100	406,000
SHR.(Vel. Set Zero)	15,500	136,200	193400
F_{ros} in [-8,10]			
IP Spread Dist.	302,300(28)	774,900	995,000
IP Confined Dist.	296,600(32)	729,000	955,000
Exponential Spread Dist.	208,800(24)	754,700	985,200
Exponential Confined Dist.	301,100(33)	801,400	961,800
Periodic(Vel. Recomputed)	26,200(27)	705,100	986,200
Periodic(Vel. Unchanged)	247,300(32)	776,800	994,900
Random(Vel. Recomputed)	311,200(30)	809,300	990,800
Random(Vel. Unchanged)	380,100(29)	793,300	968,300
SetOnBoundary(Vel. Recomputed)	248,700(35)	795,600	973,900
SetOnBoundary(Vel. Reflected)	661,900(1)	661,900	661,900
SetOnBoundary(Vel. Set Zero)	117,400(25)	858,400	995,400
SHR.(Vel. Recomputed)	347,900(33)	790,500	996,300
SHR.(Vel. Set Zero)	353,300(26)	788,700	986,800
F_{ros} in [1,10]			
Strategy	**Best**	**Median**	**Worst**
IP Spread Dist.	184,600(47)	442,200	767,500
IP Confined Dist.	229,900(40)	457,600	899,200
Exponential Spread Dist.	19,400(47)	378,200	537,300
Exponential Confined Dist.	*6.79e-03 (DNC)*	*4.23e+00 (DNC)*	*6.73e+01 (DNC)*
Periodic(Vel. Recomputed)	*1.51e-02 (DNC)*	*3.73e+00 (DNC)*	*5.17e+02 (DNC)*
Periodic(Vel. Unchanged)	*1.92e+04 (DNC)*	*2.86e+04 (DNC)*	*6.71e+04(DNC)*
Random(Vel. Recomputed)	103,800	432,200	527,200
Random(Vel. Unchanged)	*2.33e+02 (DNC)*	*1.47e+03 (DNC)*	*4.23e+03 (DNC)*
SetOnBoundary(Vel. Recomputed)	*1.71e+01 (DNC)*	*1.87e+01 (DNC)*	*3.13e+02 (DNC)*
SetOnBoundary(Vel. Reflected)	*6.88e+00 (DNC)*	*5.52e+02 (DNC)*	*2.14e+04 (DNC)*
SetOnBoundary(Vel. Set Zero)	*6.23e+00 (DNC)*	*1.80e+01 (DNC)*	*3.12e+02 (DNC)*
SHR.(Vel. Recomputed)	350,300(3)	350,900	458,400
SHR.(Vel. Set Zero)	163,700(26)	418,000	531,900