Experimental Study on Bound Handling Techniques for Multi-Objective Particle Swarm Optimization

Devang Agarwal and Deepak Sharma

Department of Mechanical Engineering,
Indian Institute of Technology Guwahati,
Guwahati, Assam, India, PIN-781039
Email: {a.devang, dsharma}@iitg.ernet.in

Abstract

Many real world optimization scenarios impose certain limitations, in terms of constraints and bounds, on various factors affecting the problem. In this paper we formulate several methods for bound handling of decision variables involved in solving a multi-objective optimization problem using particle swarm optimization algorithm. We further compare the performance of these methods on different 2-objective test problems.

Keywords: Bound Handling, Multi-objective optimization, Particle Swarm Optimization

I. Introduction

Particle swarm optimization (PSO) is a stochastic, population-based evolutionary algorithm inspired by the collective behavior of flocks and uses swarm intelligence to perform the task of search and optimization [1]. The process starts with random particles (points lying between the specified upper and lower bounds) being generated in the search space. The particles move in the search space guided by their own experiences and the acquired knowledge of the swarm, i.e., they have personal and global guides that guide them to better regions (solutions) in the solution space. These guides are the best solutions discovered so far by the particle itself, in the case of the personal guide, and the best solutions discovered so far collectively by the group in the case of the global guides. In the latter case each particle is assigned a global guide based on some criteria. When we say the particles move we essentially mean to say that the position of the particle is changed or updated after every iteration. A velocity vector is added to the current position vector to move it to a new position. The personal guides and global guides play a major role in the formulation of the velocity vector. One of the biggest problems faced with PSO is that of maintaining the swarm within the feasible region, especially when swarm goes out the bound [2, 3]. The particles get large velocities. The velocity vector adds momentum to the particle and forces the particle out of the feasible region. Empirical analysis and theoretical proofs show that particles leave search boundaries very early during the optimization process [4, 5]. This results in inefficient performance of the algorithm, and poor convergence of the swarm, as even after millions of function evaluations, the swarm fails to converge, wasting a lot of time. As a result, in most cases, we do not get any feasible solution at all.

Over the past, several methods have been proposed in the literature to extend PSO to deal with multi-objective optimization problems, which are known as Multi-Objective Particle Swarm Optimization (MOPSO) techniques [6, 7, 8]. However so far, little attention has been drawn for bound handling to MOPSO [9]. A task of respecting the bounds is even more difficult with MOPSO when many particles are likely to converge to the Pareto-optimal front and get out of the bounds. Helwig et al. in [2] have suggested several methods for bound handling on a flat landscape. Padhye et al. [3] suggested inverse parabolic distribution strategy and compared the results with other techniques for single-objective optimization. In this paper we implement the various methods of [2] for the bound handling in multi-objective problems. These methods can help to not only achieve solutions faster but also can help improving the efficiency of PSO. We test the methods on benchmark problems including ZDT1, ZDT2, ZDT3 and ZDT6 [10]. It is observed that the swarm fails to converge to the Pareto-optimal front in the absence of coupling with one of the bound handling methods mentioned in [2].

The outline of the paper is as follows. In section II we describe the general scheme of the PSO algorithm. In section III we describe the bound handling methods that we have coupled with MOPSO. In section IV we present the results of various bound handling methods with MOPSO for ZDT1, ZDT2, ZDT3 and ZDT6 benchmark multi-objective problems. Finally in section V we draw our conclusion.

II. Generalized MOPSO Algorithm

The generalized MOPSO algorithm is given in table 1. It is a simple MOPSO algorithm coupled with a boundary handling technique. The fitness is assigned to each particle using non-dominated sorting and crowding distance operators [11]. The following equations illustrate the position, velocity, personal guide and global guide updates respectively.

(1) Position of the particle (i) is adjusted as- $x_i^{t+1} = x_i^t + v_i^{t+1}$

(2) Velocity of particle (i) is updated as-

$$v_i^{t+1} = wv_i^t + c_1 r_1 \left(p_{i,lb}^t - x_i^t\right) + c_2 r_2 \left(p_{i,gb}^t - x_i^t\right)$$

Where i is the i^{th} particle, t is the iteration number, v_i^0 is set randomly, w adds to inertia of particle, n is the number of particles in the swarm, c_1 and c_2 are acceleration coefficients, and r_1 and r_2 are random numbers ∈ [0, 1].

(3) $p_{i,lb}^t$ is the personal guide of the i^{th} particle at t^{th} iteration which is calculated as-

If $f(x_i^{t+1})$ has better fitness than $f(p_{i,lb}^t)$ $then$ $p_{i,lb}^{t+1} = x_i^{t+1}$

Else $p_{i,lb}^{t+1} = p_{i,lb}^t$

(4) $p_{i,gb}^t$is the global guide of the i^{th} particle at t^{th} iteration, calculated as-

$$p_{gb}^t \in \{p_{1,lb}^t, p_{2,lb}^t, .., p_{n,lb}^t\} \mid f\ (p_{gb}^t) = best\{f(p_{1,lb}^t), f(p_{2,lb}^t), .., f(p_{n,lb}^t)\}.$$

The general MOPSO scheme can be modified to produce improved results and faster convergence of the swarm, by introducing the concept of bound handling while updating the position of the particle. This maintains all the particles within the feasible search region always during the search. These bound handling methods are described in the following section.

III. Bound Handling Methods

In this section we discuss various bound handling methods which are coupled with MOPSO.

Table 1

MOPSO coupled with Boundary Handling Technique

```
1. Set the iteration counter, t =0, maximum allowed iterations = T, c1 = 1.7, c2 = 1.6 and w = 0.9.
2. Initialize position and velocity of the particles randomly.
3. Evaluate particles and assign fitness.
4. while t < T do
   a. Calculate $p_{i,lb}^t$using equation (3).
   b. Calculate $p_{i,gb}^t$using equation (4).
   c. Update velocity for each particle using equation (2).
   d. Update position for each particle using equation (1).
   e. Check If the position of the particle has exceeded the bounds –
      If yes,
      Using a boundary handling technique described in section III, update the position and velocity for each particle.
      Else,
      Keep the same position and velocity as calculated in 4(b) and 4(c).
   f. Evaluate particles and assign fitness.
   g. t = t + 1
5. end while
```

1. **Mutation**: Mutation based method that perturbs the unbounded position by a small amount to bring the particle back into the feasible region.
2. **Reflect Methods**: In the reflect method we reflect all the violating variables along the closer bound and repeat the process unless the variable is within the bounds. There are three variations to alter the velocity.
    i. **Reflect-Adjust:** $V_{new} = X_{new,bounded} - X_{old,unbounded}$
    ii. **Reflect Unmodified:** Velocity is unmodified.
    iii. **Reflect-Zero**: Velocity is set to zero.
3. **Nearest Methods**: Put the particle on the nearer bounds with four variations to alter the velocity.
    i. **Nearest–Zero**: Velocity is set to zero.
    ii. **Nearest–Unmodified**: Velocity is unmodified.
    iii. **Nearest with deterministic back** : $V_{new} = -0.5 * V_{old}$
    iv. **Nearest with random back**: $V_{new} = -lambda * V_{old}$, where lambda is a random real number between 0 and 1.
4. **Random Methods**: Randomly locate the particle within the bounds with two variations to change the velocity.
    i. **Random-Zero:** Velocity is set to zero.
    ii. **Random-Unmodified:** Velocity is unmodified.
5. **Hyperbolic Method**: In this method the particle is not allowed to leave the feasible area at all. The new velocity is normalized before updating the position as,
   If $V_i^{t+1} > 0$ $\qquad V_i^{t+1} = V_i^{t+1} / \left(1 + \left|V_i^{t+1} / \left(X_{i,max} - X_i^t\right)\right|\right)$
   Else $\qquad V_i^{t+1} = V_i^{t+1} / \left(1 + \left|V_i^{t+1} / \left(X_i^t - X_{i,min}\right)\right|\right)$
   The normalized velocity ensures that the new position of the particle is always within the bounds.

The methods described in this section help to handle variable bounds during the execution of the algorithm. These methods restrict the movement of the particle thereby keeping them in the feasible region. Hence they help the algorithm to converge faster and find better solutions. We now describe the benchmark problems over which we have tested the MOPSO algorithm coupled with these methods.

IV. Results and Discussion

MOPSO with various bound handling methods is tested on four ZDT benchmark problems [10] in multi-objective optimization. The mathematical description of the problems is given in table 2 with their nature of Pareto-front. Their characteristic features of convexity, non-convexity, discreteness and non-uniformity respectively are known to cause difficulty to an EA in converging to the Pareto optimal front. We run our code 20 times with different initial particle positions using the algorithm described in table 1. The performance of bound handling methods are assessed by using inverse generalized distance. The best, median and worst inversed generalized distance (IGD) values [12] from our runs are tabulated in table 3.

For solving problems with convex Pareto fronts, the best performance is shown by the Hyperbolic method, but it is not consistent, hence Nearest Z serves as the best method for such problems as shown by its consistent IGD values for ZDT1 problem. Other versions of the Nearest method can also prove to be useful.

Solving problems having a non-convex front like ZDT2 proves to be difficult using MOPSO even after the deployment of bound handling methods. Although Nearest Z shows maximum convergence in the best case in ZDT2 problem, but in several other runs it fails to live up to its best performance. Similar behavior is shown by Nearest U and Nearest DB methods. However Nearest RB shows incredible consistency in solving ZDT2 problem as evident from the consistent best, median and worst IGD values for ZDT2 problem. Hence for solving problems with a previous knowledge of it having a non-convex Pareto front, Nearest RB method could prove to be extremely useful for an improved bound handling and a better convergence to the true Pareto using MOPSO. None of the other methods solve the ZDT2 problem.

When there are several non-contiguous convex parts in the Pareto front, both Nearest U and Nearest Z perform the best and equally well as in the case of ZDT3 problem. Hyperbolic and other versions of Nearest method could also be used as they show consistent IGD values for all cases. Reflect Z and A also may be used, while other methods result in weak Pareto front.

In the case of ZDT6 problem which has a non-uniformly distributed, non-convex Pareto-optimal front, satisfactory performance is shown by Hyperbolic, all Nearest methods and Reflect A, Z methods. Nearest RB method is consistent and shows the best performance in the worst case.

The analysis of table 3 and the plots in figure 1 give a clear idea that out of all the methods, Random-U, Random-Z and Reflect U are the poorest and generate weak Pareto fronts in all cases. The reason for the poor performance of Reflect U is that even after being reflected back into the feasible region, the particle is forced out again into infeasibility because of the unmodified velocity vector. The Random methods also fail because they randomly put the violating variable in the feasible region. In both the above cases even though feasibility is maintained, the swarm loses out on the areas of better fitness. Hence the particles fail to converge to the Pareto front. The other two variants of Reflect method, Reflect-A and Reflect-Z perform much better than their counterpart Reflect U because in these two cases the velocity vector is modified (reversed or made zero) which helps the particle to remain in the feasible region and eventually reach near the true Pareto front. However the results acquired with them still have a scope of further improvement. They may be used (although some other methods are much better) to solve problems with uniform or discrete convex fronts like ZDT1 and ZDT3 and non-uniform problems like ZDT6 but fail miserably in solving non-convex problems like ZDT2 problem. (See the IGD values for ZDT2). The mutation based method also fails at a general level in the context of the shape of the Pareto front, poor convergence and getting stuck in the local optimum. One straight forward explanation is that the small perturbation from the unbounded position does not ensure feasibility. Hyperbolic, though not the best, can still be used with all problems mentioned in the paper except ZDT2. Its performance can be attributed to the fact that it never allows the particle to leave the feasible region and hence has proven to be quite competitive. As mentioned in [2] the velocities when using Hyperbolic are very small, which reduces the exploration capability of the swarm and it is highly probable that it prematurely converges at a local optimum. This explains its poor

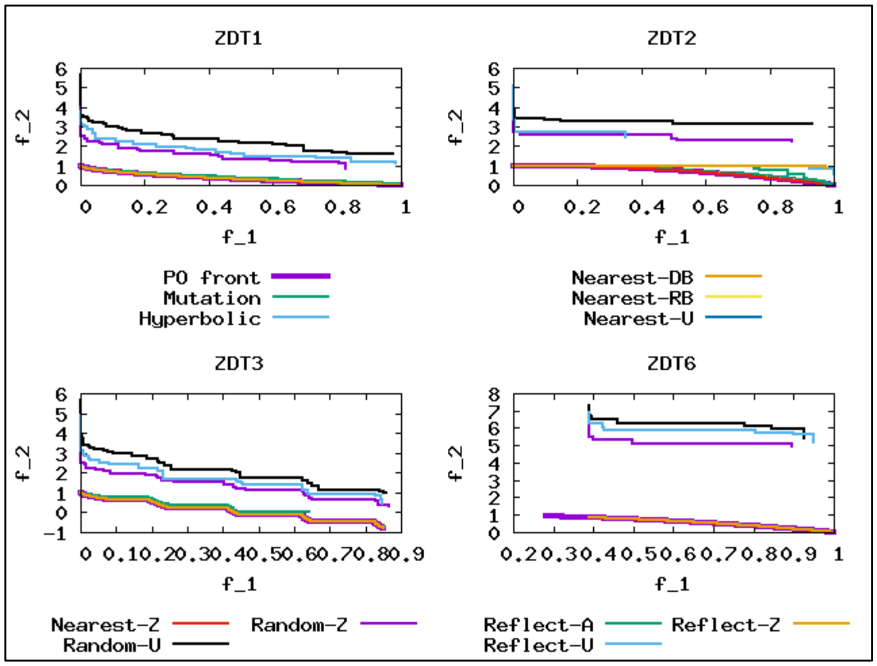

Figure 1: 0% attainment plots for ZDT problems using MOPSO with bound handling techniques

performance in solving ZDT2 problem.
All the Nearest methods have performed well. Nearest RB method has proven to be the most consistent. The method never fails as depicted by the IGD values which are the best or very close to the best compared to other methods even in the worst cases.
It is closely followed by Nearest Z and Nearest DB methods that also give good results most of the time. Nearest U method does not perform as good as the rest of them due to the unmodified velocity vector.
An important reason to be noted here, which explains the superior performance of all the Nearest methods is that with ZDT problems, part of the Pareto front lies at the bound, and since the Nearest method keeps the particle at the bound to prevent infeasibility, often the particles are able to reach a local optimum and then they diverge from there to form the Pareto front. Nearest RB performs the best, as it sets the velocity randomly compared to the other Nearest versions where the velocity is predetermined which limits the exploration capability of the swarm.
From our analysis in this section we can say that the Nearest Methods and to some extent the Hyperbolic method, are the most reliable methods for bound handling. Finally we have identified that the Nearest RB method, may not produce the best results in all cases, but it is very close to the best in all the cases. Hence we recommend that, Nearest RB method should be used for bound handling with various problem types.

V. Conclusion

In this paper we implemented several methods for bound handling of decision variables for multi-objective particle swarm optimization, and showed that MOPSO coupled with boundary handling techniques performs better than without the usage of a well-defined boundary handling technique. The Nearest methods and the Hyperbolic method perform well at handling the bounds, particularly the Nearest RB method beats them all. The reflect method also gives fair results. Mutation based method disappoints given its popularity in the literature. The algorithm works almost perfectly for most of the 2-D benchmark problems including ZDT1, ZDT2, ZDT3 and ZDT6. The future scope of research and study lies in the area of development of further efficient bound handling methods for MOPSO that can be used for multi and many objective optimization problems.

References

[1] Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In Proceedings of IEEE International Conference on Neural Networks. **4**, 1942–1948 (1995).

[2] Helwig, S., Branke, J., Mostaghim, S.: Experimental Analysis of Bound Handling Techniques in Particle Swarm Optimization. IEEE transactions on Evolutionary Computation. **17**(2), 259–271 (2013).

[3] Padhye, N., Deb, K., Mittal, P.: Boundary Handling Approaches in Particle Swarm Optimization. In Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012), Advances in Intelligent Systems and Computing. **201**, 287–298 (2013).

[4] Helwig, S., Wanka, R.: Particle Swarm Optimization in High Dimensional Bounded Search Spaces. In Proceedings IEEE Swarm Intelligence Symposium, 198–205 (2007).

[5] Helwig, S., Wanka, R.: Theoretical Analysis of Initial Particle Swarm Behavior. In Proceedings of 10th International Conference PPSN, 889–898 (2008).

[6] Coello, C.A.C., Pulido, G.T., Lechuga, M.S.: Handling Multiple Objectives with Particle Swarm Optimization. IEEE Transactions on Evolutionary Computation. **8**(3), 256–279 (2004).

[7] Wang, Y., Li, B., Weise, T., Wang, J., Yuan, B., Tian, Q.: Self-Adaptive Learning Based Particle Swarm Optimization. Information Sciences. **181**(20), 4515–4538 (2011).

[8] Li, F., Xie, S., Ni, Q,: A Novel Boundary Based Multiobjective Particle Swarm Optimization. In Advances in Swarm and Computational Intelligence. **9140**, 153–163 (2015).

[9] Padhye, N., Branke, J., Mostaghim, S.: Empirical Comparison of MOSPO Methods - Guide Selection and Diversity Preservation -. In Proceedings of Congress of Evolutionary Computation. 2516–2523 (2009)

[10] Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation, **8**(2), 173–195 (2000).

[11] Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation. **6**(2), 182–197 (2002).

[12] Jenkins, W.K., Mather, B., Munson, D.C., Jr.: Nearest Neighbor and Generalized Inverse Distance Interpolation for Fourier Domain Image Reconstruction. In Acoustics, Speech, and Signal Processing, IEEE International Conference on ICASSP '85. , **10**, 1069–1072 (1985).

Table 2: Details of ZDT Benchmark Problems [10]

	n	Variable bounds	Objective Functions	Optimal Solutions	Nature Of Pareto front
ZDT1	30	[0 , 1]	$f_1(x_1) = x_1$ $g(x_2,\dots,x_m) = 1 + 9.\sum_{i=2}^{m} x_i/m - 1$ $h(f_1,g) = 1 - (f_1/g)^2$	$x_1 \in [0,1]$ $x_i = 0,$ $i = 2,\dots,$ n	Convex Continuous
ZDT2	30	[0 , 1]	$f_1(x_1) = x_1$ $g(x_2,\dots,x_m) = 1 + 9.\sum_{i=2}^{m} x_i/m - 1$ $h(f_1,g) = 1 - \sqrt{f_1/g}$	$x_1 \in [0,1]$ $x_i = 0,$ $i = 2,\dots,$ n	Nonconvex Continuous
ZDT3	30	[0 , 1]	$f_1(x_1) = x_1$ $g(x_2,\dots,x_m) = 1 + 9.\sum_{i=2}^{m} x_i/m - 1$ $h(f_1,g) = 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\Pi f_1)$	$x_1 \in [0,1]$ $x_i = 0,$ $i = 2,\dots,$ n	Convex Disconnected
ZDT6	10	[0 , 1]	$f_1(x_1) = 1 - \exp(-4x_1) sin^6(6\Pi x_1)$ $g(x_2,\dots,x_m) = 1 + 9.\left(\left(\sum_{i=2}^{m} x_i\right)/(m-1)\right)^{0.25}$ $h(f_1,g) = 1 - (f_1/g)^2$	$x_1 \in [0,1]$ $x_i = 0,$ $i = 2,\dots,$ n	Nonconvex non uniformly spaced
All the problems are minimized.					

Table 3: Best, Median and Worst IGD values obtained for MOPSO. Best performance is shown in bold.

	Mutation	Hyperbolic	Nearest-DB	Nearest-RB	Nearest-U	Nearest-Z	Random-U	Random-Z	Reflect-A	Reflect-U	Reflect-Z
ZDT1	5.414e-03	**4.427e-04**	6.083e-04	5.656e-04	4.502e-04	4.795e-04	1.311e-01	7.272e-02	8.593e-04	9.585e-02	7.337e-04
	1.251e-02	5.382e-04	6.681e-04	6.577e-04	**4.913e-04**	5.404e-04	1.489e-01	8.971e-02	1.170e-03	1.109e-01	9.854e-04
	2.238e-02	7.458e-02	8.004e-04	7.284e-04	8.695e-04	**5.795e-04**	1.692e-01	9.654e-02	3.095e-03	1.244e-01	6.538e-03
ZDT2	4.838e-03	3.409e-02	4.876e-04	4.518e-04	4.223e-04	**4.162e-04**	2.285e-01	1.508e-01	1.776e-02	1.686e-01	4.395e-02
	1.398e-02	7.231e-02	5.736e-04	**5.531e-04**	1.647e-03	5.462e-02	2.468e-01	1.944e-01	7.231e-02	2.013e-01	7.231e-02
	2.036e-02	8.918e-02	7.231e-02	**6.085e-04**	7.231e-02	7.231e-02	2.636e-01	2.099e-01	8.918e-02	2.173e-01	7.332e-02
ZDT3	1.886e-02	3.697e-04	3.933e-04	3.587e-04	3.447e-04	**3.423e-04**	7.916e-02	5.120e-02	6.716e-04	6.136e-02	5.431e-04
	2.005e-02	4.124e-04	4.252e-04	4.202e-04	**3.856e-04**	4.106e-04	8.515e-02	5.493e-02	1.471e-03	6.876e-02	9.870e-04
	2.822e-02	6.612e-04	4.579e-04	5.301e-04	4.680e-04	**4.461e-04**	9.164e-02	6.162e-02	2.917e-03	7.519e-02	3.744e-03
ZDT6	3.508e-03	3.174e-03	**3.165e-03**	3.194e-03	3.498e-03	3.503e-03	5.367e-01	4.848e-01	3.301e-03	5.075e-01	3.197e-03
	2.271e-01	3.489e-03	3.523e-03	**3.476e-03**	3.506e-03	3.509e-03	6.221e-01	5.498e-01	4.055e-03	5.731e-01	3.503e-03
	3.187e-01	3.532e-03	1.336e-02	**3.515e-03**	3.519e-03	3.524e-03	6.489e-01	5.787e-01	7.089e-03	6.019e-01	4.309e-03

DB - Deterministic Back, RB - Random Back, Z - Zero, A - Adjust and U - Unmodified.