Evaluation of Penalty Function Methods for Constrained Optimization Using Particle Swarm Optimization

L. Ashoka Vardhan
Former UG Student, BITS Pilani – Hyderabad Campus
Jawahar Nagar, Shameerpet Mandal
Hyderabad 500 078, India
ashok.lella@gmail.com

Arunachalam Vasan
Associate Professor, BITS Pilani – Hyderabad Campus
Jawahar Nagar, Shameerpet Mandal
Hyderabad 500 078, India
vasan@hyderabad.bits-pilani.ac.in

***Abstract*— Solving complex problems with higher dimensions involving many constraints is often a very challenging task. While solving multidimensional problems with particle swarm optimization involving several constraint factors, the penalty function approach is widely used. This paper provides a comprehensive survey of some of the frequently used constraint handling techniques currently used with particle swarm optimization. In this paper some of the penalty functional approaches for solving evolutionary algorithms are discussed and a comparative study is being performed with respect to various benchmark problems to assess their performance.**

Keywords—penalty function approach; particle swarm optimization; evolutionary algorithms; swarm intelligence

I. INTRODUCTION

Evolutionary algorithms (EA) compared to traditional algorithms have become quite popular in the last decade due to successful implementation in various applications across the disciplines. These algorithms when compared to conventional algorithms are inspired by nature's evolution techniques to evolve to better solutions from the available search space. Unlike conventional algorithms they converge down to different solutions for different runs. These algorithms are generally designed to work for unconstrained problems. But in reality, most of the real world planning problem is constrained in nature. Therefore, it is necessary to use an efficient constraint handling technique to convert the constrained optimization problem into an unconstrained optimization problem. Penalty function approach is one of the most famous techniques of constraint handling.

The working of the evolutionary algorithm begins by creating a set of population of individuals. The initialization of each individual is done randomly between the lower and upper bound of a variable which represents a possible solution to the optimization problem. The fitness value of an individual is determined by substituting this set of values to the formulated objective function of the problem for every iteration. If the nature of the objective function is to minimize, the individuals having lower fitness value compared to the rest are mutated. The mutated individuals might provide better results compared to the rest of the set. The process is iterated till all the individuals get mutated over time to converge to the optimal solution.

II. PENALTY FUNCTIONS

Penalty functions are used when dealing with constrained optimization problems. Penalties are the most common approach in the evolutionary algorithm problems to handle constraints, especially to solve the ones including inequality constraints. Courant R. [1] was the first to propose penalty functions in 1940's and later they were expanded by Fiacco & Mc-Cormick [2]. Their main idea was to build an unconstrained optimization problem from an ordinary constrained optimization problem by modifying the objective function using weights based on the amount of constraint violation present in a certain solution.

Penalty functions have continued to exist in literature of constrained optimization for decades. Based on the degree where penalty is applied, penalty functions can be broadly classified into three categories [3].

1. Partial penalty functions: Penalty is applied near boundary of feasibility
2. Global penalty functions: Penalty is applied throughout the infeasible region
3. Barrier methods: No feasible solution is considered.

The popular Lagrangian relaxation method [4] in the field of combinatorial optimization uses the same theme by relaxing the most difficult constraints temporarily and then in order to avoid straying too far for the feasible region, it uses a modified objective function.

Any penalty function in general pursues the following pattern. Given an optimization problem,

978-1-4673-6101-9/13/$31.00 ©2013 IEEE

$$\begin{aligned} &Min \quad f(x) \\ &s.t \quad x \in A \\ &\qquad x \in B \end{aligned} \tag{1}$$

Where, **x** is a vector of decision variables, here in between the constraint sets **A** and **B,** constraints of that of **B** are relatively difficult to satisfy. Thus the problem can be reformulated as,

$$\begin{aligned} &Min \quad f(x) + p(d(x.B))(R) \\ &s.t \quad x \in A \end{aligned} \tag{2}$$

where,

d(**x**,B) is a metric function stating the distance between the solution vector **x** and the region B,

p(.) is a monotonically non-decreasing penalty function for which p(0)=0.

If p(.) grows quickly enough outside of B, the optimal solution of (P) will also be optimal for (R). Furthermore, any optimal solution of (R) will provide an upper bound on the optimum for (P), and this bound will in general be tighter than that obtained by simply optimizing f(x) over A.

Usually, the second constraint of equation (1) which is relatively difficult to satisfy is expressed as an inequality and constraints in the form

$$\begin{aligned} &g(x) \le 0 \quad for \quad i = 1.....q \\ &h(x) = 0 \quad for \quad i = q+1.....m \end{aligned} \tag{3}$$

where, q = number of inequality constraints; m-q = number of equality constraints.

In classical optimization, two kinds of penalty functions are considered: exterior and interior. Exterior penalty functions are used for penalizing infeasible solutions while internal penalty functions are used for penalizing feasible solutions i.e., in the case of exterior methods, the optimization process is initiated outside the feasible region and later is made to move towards a feasible region. Whereas in case of an internal penalty function, the initial penalty term is chosen such that its value is increased to a very large value so that the points move away from the constraint boundaries. If started form a feasible region, the points are forced to lie within the feasible region during the optimization process as the constraint boundaries are acting as barriers.

Exterior penalty approach is chosen by most of the researchers in EA community because of their nature to solve optimization problems without requiring an initial feasible solution. In general, it is very difficult to generate a feasible solution in the initial phase which is a main drawback in using interior penalties as many of the problems in which EAs are intended, the problem of finding a feasible solution is itself.

It can be difficult to find a penalty function which is capable of solving a given optimization problem efficiently. Also, a sensitivity analysis has to be done to determine the penalty function for a given problem to ensure that the direction of search ensures to maintain the eventual solution quality. Moreover, much of the difficulty arises because the optimal solution will frequently lie on the boundary of the feasible region. One has to be very careful, as most of the solutions that are similar compared to that of the optimum solution might be infeasible. Thus, limiting the search to only feasible solutions makes it difficult to find the schemata for optimizing the population to move towards the solution space. But, however if the region is explored, much of the search time will be used for exploring the regions far from the feasible region.

A. Types of penalty functions

Death Penalty:

This is the most basic and popular method of penalty approach which simply rejects all unfeasible solutions from the population domain.

$$P(\mathbf{x}) = +\infty, \qquad \mathbf{x} \in \mathbf{S-F} \tag{4}$$

In this method, the process always succeeds to never produce any unfeasible solutions in the population. In a search space of convex domain or a reasonable part of the whole workspace, this function can be expected to work well. However, in case of abstruse situations where the problem is highly convoluted, the algorithm runs relatively slower, spending much time to land at a handful of feasible solutions. Also, considering only the points that are in feasible region of the search space prevents to find better solutions.

Dynamic Penalty:

Joines and Houck [5] proposed a technique in which the population is evaluated using the below mentioned fitness function

$$fitness(x) = f(x) + (Cxt)^{\alpha} \times SVC(\beta, x) \tag{5}$$

Where, α, β and C are constants defined by the user (the authors used C=0.5, β=2.0 or 1.0 and α=2.0 or 1.0), and SVC (β,**x**) is defined as [5]:

$$SVC(\beta, x) = \sum_{i=1}^{n} D_i^{\beta}(x) + \sum_{j=1}^{p} D_j(x) \tag{6}$$

and

$$D_i(x) = \begin{cases} 0 & g_i(x) \le 0 \\ |g_i(x)| & otherwise \end{cases} \qquad 1 \le i \le n \tag{7}$$

$$D_i(x) = \begin{cases} 0 & -\in \le h_j \le \in \\ |h_j(x)| & otherwise \end{cases} \qquad 1 \le j \le p \qquad (8)$$

The penalty value of the dynamic function illustrated above increases gradually as we progress through generations.

Kazarlis and Petridis [6] performed a detailed study of the behaviour of dynamic penalty functions of the form:

$$fitness(x) = f(x) + V(g) \times (A\sum_{i=1}^{m}(\delta_i w_i \varphi(d_i(s))) + B) \times \delta_i \qquad (9)$$

where,
A = "Severity" factor,
m = the total number of constraints,
δ_i = 1 if the constraint *i* is violated and 0 otherwise,
w_i = weight factor for constraint *i*,
$d_i(S)$ = measure of the degree of violation on constraint i introduced by solution S,
$\varphi_i(.)$ = function to measure degree of violation,
B = penalty threshold factor,
δ_i =binary factor (d_s=1 if S is infeasible and is zero otherwise) ,
g= current generation number
V(g) = increasing function of g in the range (0. . .1).

Using unit commitment problem and the cutting stock problem as test functions, Kazarlis and Petridis experiment with different forms of V(g) (linear, quadratic, cubic, quartic, exponential and 5-step), and found that the best overall performance was provided by a function of the form:

$$V(g) = \left(\frac{g}{G}\right)^2 \qquad (10)$$

where, G = total number of generations.

Annealing Penalty:

The name is inspired from annealing process in metallurgical sciences where the heat treatment alters a material to increase its ductility to make it more workable. Michalewicz and Attia [7] considered a method based on the idea of simulated annealing [8]: unlike other penalty function the rate of change of penalty coefficients is relatively low (after the algorithm has been trapped within local optima). Only active constraints are considered at each iteration, and the penalty is increased over time (i.e., the temperature decreases over time) resulting in heavy penalization of infeasible individuals in the last generations.

In order to apply Michalewicz and Attia's [7] penalty functions, it is required to divide the constraints into four groups:

a) linear equalities,
b) linear inequalities,
c) nonlinear equalities and
d) nonlinear inequalities.

Also, a set of active constraints A has to be created, and all nonlinear equalities together with all violated nonlinear inequalities have to be included there. The population is evolved using [7]:

$$fitness(x) = f(x) + \frac{1}{2\tau}\sum_{i \in A}\varphi_i^2(x) \qquad (11)$$

where τ is the cooling schedule [6]

$$\varphi_i(x) = \begin{cases} \max[0, g_i(x)] & if\ 1 \le i \le n \\ |h_i(x)| & if n+1 \le i \le m \end{cases} \qquad (12)$$

and, m is the total number of constraints.

Unlike other penalty functions, when simulated annealing is used, the initial population is not very diverse and it consist multiple clones of single units, satisfying the objective functions linear constraints. The temperature τ is gradually decreased at each iteration and the best solution in the previous iteration is used for creating population for next iteration. The process is halted when a pre-defined τ_f (final 'freezing' temperature) is attained.

A similar proposal is that the fitness of an individual is computed using:

$$fitness(x) = A.f(x) \qquad (13)$$

Where, A depends on two parameters: M, which, measures the amount of constraint violation, and T, which is a function of the running time of the algorithm. T tends to zero as evolution progresses. Using the basic principle of annealing A is defined as:

$$A = e^{-M/T} \qquad (14)$$

To define T, Carlson used

$$T = \frac{1}{\sqrt{t}} \qquad (15)$$

where t refers to the temperature used in the previous iteration.

III. PARTCILE SWARM OPTIMIZATION

Particle Swarm Optimization (PSO) is a population based stochastic optimization algorithm in swarm intelligence [9]. This algorithm is becoming popular due to its simplicity of implementation and ability to quickly converge to a reasonably good solution [10]. Just like other conventional EA's the initial population here is commenced with random solutions and the algorithm over generations searches for global optima. But, here PSO does not have evolutionary operations such as the crossover and mutation. Here, other solutions fly through the search space following the present iterations optimal value and the global optima found up to the present iteration. Here, potential solutions known as particles keep track of their earlier coordinates which are associated with the best solution (fitness) it has achieved so far. The pbest and lbest value i.e., fitness value and the best value of any particle obtained in the neighboring particles so far tracked during the process respectively are also stored similarly. When a particle takes all the population as its topological neighbors, the best value is a global best and is called gbest. These swarm of particles for every iteration change their velocity and acceleration towards its pbest and lbest locations where, the acceleration is weighted randomly which is specially generated for every iteration of the process. The velocity (v) and the position (x) of the ith swarm are manipulated according to the following two equations:

$$v_{ij}^{k+1} = \chi\left[\omega v_{ij}^{k} + C_1 R_1\left(p_{ij}^{k} - x_{ij}^{k}\right) + C_2 R_2\left(p_{gj}^{k} - x_{ij}^{k}\right)\right] \quad (16)$$

$$x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1} \quad (17)$$

Where,

i = number of particles

j = number of decision variables

k = iteration counter

χ = is the constriction factor which controls and constricts the magnitude of the velocity

g = gbest particle

p = pbest particle

ω = intertia weight which is often used as a parameter to control exploration and exploitation in the search space;

R1, R2 = random variables uniformly distributed within [0, 1];

C1, C2 = acceleration coefficients, also called the cognitive and social parameters respectively. C1 and C2 are popularly chosen to vary within {0, 2} [11].

The following are the termination criteria for PSO: (i) The algorithm terminates when it reaches the maximum iterations or (ii) the difference between the best solutions of last two successive iterations reached a pre-defined value.

PSO has been successfully applied in many areas of research over the past several years. It has also been demonstrated that PSO is more robust and simple as it is faster and cheaper compared to others. The best part of PSO is its ability to adjust only a few parameters while fine tuning for the solution. Altering one version with minor modifications works really well in a wide variety of applications. Particle swarm optimization has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement.

IV. RESULTS AND DISCUSSIONS

The following are the five benchmark problems on which the comparison of various penalty function approaches using PSO has been executed:

Benchmark Problem 1 (BM 1):

$$Min\, f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + {x_3}^4 + 3(x_4 - 11)^2 + 10{x_5}^6 + 7{x_6}^2 + {x_7}^4 - 4x_6x_7 - 10x_6 - 8x_7 \quad (18)$$

$$s.t. -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \le 0$$
$$-282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \le 0$$
$$-196 + 23x_1 + {x_2}^2 + 6x_6^2 - 8x_7 \le 0$$
$$4{x_1}^2 + {x_2}^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \le 0$$
$$-10 \le x_i \le 10, \quad i = 1, \ldots, 7$$

The optimum solution is x^*= (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227) where $f(x^*) = 680.6300573$

Benchmark Problem 2 (BM 2):

$$\max f(x) = -2x_1^2 + 2x_1\, x_2 - 2x_2^2 + 4x_1 + 6x_2, \quad (19)$$
$$s.t. \quad x_1 + x_2 \le 2,$$
$$x_1 + 5x_2 \le 5,$$

where $x_i \ge 0$ for (i = 1,2). The optimum solution is x^*= (2.2468, 2.3815) where $f(x^*) = 13.5908$

Benchmark Problem 3 (BM 3):

$$\max f(x) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2, \quad (20)$$
$$s.t. \quad (x_1 - 0.05)^2 + (x_2 - 2.5)^2 - 4.84 \le 0$$
$$4.84 - x_1^2 - (x_2 - 2.5)^2 \le 0$$

Where $0 \le x_1 \le 6$, $0 \le x_2 \le 6$. The optimum solution is x^*= (1.1284, 0.7443) where $f(x^*) = 7.1612$

Benchmark Problem 4 (BM 4):

$$\min f(x) = x_1^2 + (x_2 - 1)^2, \quad (21)$$
$$s.t. \quad x_2 - x_1^2 = 0,$$

Where $-1 \le x_1 \le 1$, and $-1 \le x_2 \le 1$. The optimum solution is x^*= $(\pm\sqrt{2}/2, 0.5)$ where $f(x^*) = .75$

Benchmark Problem 5 (BM 5):

$$\max f(x) = [sin^3(2\pi x_1)sin(2\pi x_2)]/[x_1^3(x_1 + x_2)] \quad (22)$$
$$s.t. \quad x_1^2 - x_2 + 1 \le 0,$$
$$1 - x_1 + (x_2 - 4)^2 \le 0,$$

Where $0 \le x_1, \; x_2 \le 10$. The optimum solution is x^*= $(1.22797, 4.24537)$ where $f(x^*) = 0.09583$

Table 1 shown below summarizes the results of the five benchmark problems. PFA 1 means Death Penalty, PFA 2 means Dynamic Penalty and PFA 3 means Annealing Penalty. The parameters considered for running the PSO algorithm are number of Iterations: 500, Particle Size: 100, C1=0.2 and C2=0.4.

TABLE 1. SUMMARY OF PSO RESULTS ON VARIOUS BENCHMARK PROBLEMS USING DIFFERENT PENALTY FUNCTIONS

Problem	**Optimal Solution**	**-**	**PFA 1**	**PFA 2**	**PFA 3**
BM 1	680.63	Best	6.80.66	680.68	680.67
		Mean	**685.34**	686.00	685.69
		Worst	942.01	936.19	103.05
BM 2	7.1612	Best	7.1613	7.1613	7.1613
		Mean	7.1501	**7.1584**	7.1531
		Worst	3.6723	4.9520	3.5925
BM 3	13.5908	Best	13.5908	13.5908	13.5908
		Mean	13.5916	**13.5916**	13.5916
		Worst	13.6018	13.5980	13.6101
BM 4	0.75	Best	0.7499	0.7499	0.7499
		Mean	0.7511	0.7531	**0.7505**
		Worst	0.9643	0.9757	0.8954
BM 5	0.09583	Best	0.09582	0.09582	0.09582
		Mean	0.09542	**0.095685**	0.09542
		Worst	0.02914	0.02580	0.02581

It is important to note from the best mean values for all the benchmark problems, dynamic penalty function has a better mean value of 3 out of 5 problems. There isn't much variation in its mean values for the other two problems as well. It is clearly evident that the given benchmark problems, dynamic penalty function is performing better compared to the other two penalty function approaches. In addition, the number of function evaluations for obtaining the optimum solutions for all the five problems were analyzed. It has been observed that dynamic penalty function approach converged to the optimal value with least number of function evaluations in the three out of five problems.

The convergence graphs for BM1 using the three penalty function approaches is shown below:

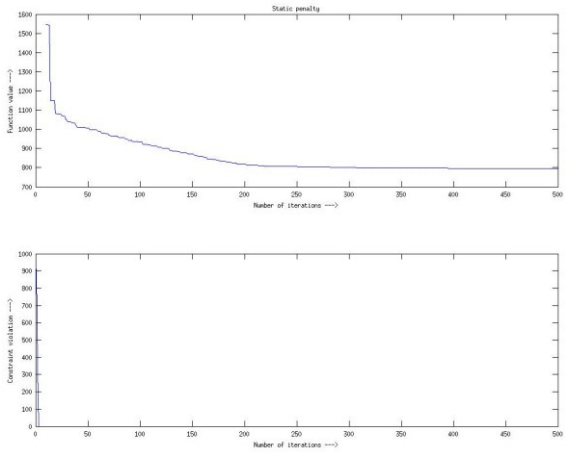

Fig. 1. Convergence Graph for BM1 using PFA1

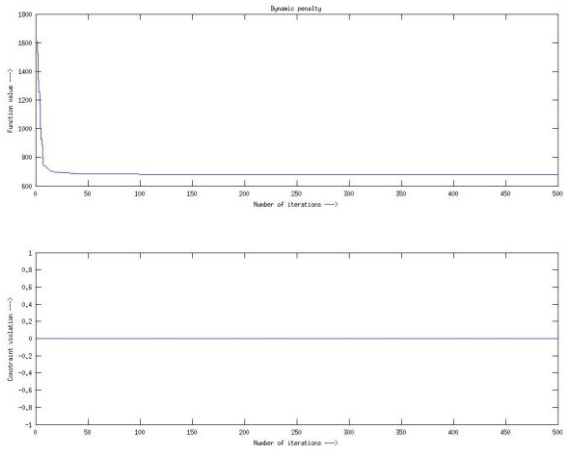

Fig. 2. Convergence Graph for BM1 using PFA2

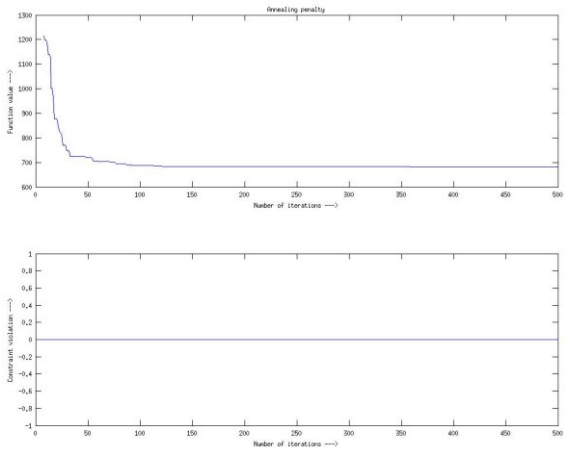

Fig. 3. Convergence Graph for BM1 using PFA3

V. CONCLUSIONS

The paper presented PSO algorithm for constrained optimization using different approaches. Dynamic penalty functions provide faster and consistent results compared to static and annealing penalty function.

ACKNOWLEDGMENT

The authors acknowledge the financial support provided by Council of Scientific and Industrial Research (CSIR) through project No. 25(0193)/11/EMR-II dated 02.02.2011.

REFERENCES

[1] R. Courant, "Variational methods for the solution of problems of equilibrium and vibrations," IBulletin of the American Mathematical Society, 49, 1-23, 1943.

[2] A.V. Fiacco, and G.P. McCormick, Nonlinear Programming: Sequential Unconstrained Minimization Techniques, John Wiley & Sons, New York, 1968.

[3] H.P. Schwefel, Evolution and Optimum Seeking, John Wiley, 1995.

[4] C.R. Reeves, Modern Heuristic Techniques for Combinatorial Problems, John Wiley & Sons, New York, 1993.

[5] J. Joines and C. Houck, "On the use f non-stationary penalty function to solve nonlinear constrained optimization problems with Gas," David Fogel, Editor, Proceedings of the First IEEE Conference on Evolutionary Computation, pp. 579-584, Orlando, Florida, 1994.

[6] S. Kazarlis and V. Petridis, Varying fitness functions in genetic algorithms: studying the rate of increase of the dynamic penalty terms, In A. E. Eibi, T. Bäck, M. Schoenauer, and H.-P. Schwefel, Editors, Parallel Problem Solving from Nature V—PPSN V, Amsterdam, The Netherlands, Springer-Verlag, 1998.

[7] Zbigneiw Michalewicz, "A survey on constraint handling techniques in computation methods," In J. R. McDonnel, R. G. Reynolds, and D. B. Fogel, editors, Proceedings of the 4th Annual Conference on Evolutionary Programming, pp. 135-155. The MIT Press, Cambridge, Massachusetts, 1995.

[8] S. Kirkpatrick, Jr. C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," Science, 220, 671—680, 1983.

[9] J. Kennedy and R.C. Eberhart, Swarm Intelligence, Morgan Kaufmann, San Mateo, California, USA, 2001.

[10] Y. Shi, and R.C. Eberhart, "A modified particle swarm optimizer," Proceedings of IEEE International Conference on Evolutionary Computation, 69-73, 1998.

[11] A. Chatterjee, and P. Siarry, "Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization," Computers and Operations Research, 33, 859-871, 2006.