Solving Constrained Nonlinear Optimization Problems with Particle Swarm Optimization

Xiaohui Hu[1, 2] and Russell Eberhart[2]

[1] Department of Biomedical Engineering
Purdue University, West Lafayette, IN, USA
hux@ecn.purdue.edu

[2] Department of Electrical and Computer Engineering
Indiana University Purdue University at Indianapolis, Indianapolis, IN, USA
reberhar@iupui.edu

ABSTRACT

This paper presents a Particle Swarm Optimization (PSO) algorithm for constrained nonlinear optimization problems. In PSO, the potential solutions, called particles, are "flown" through the problem space by learning from the current optimal particle and its own memory. In this paper, preserving feasibility strategy is employed to deal with constraints. PSO is started with a group of feasible solutions and a feasibility function is used to check if the new explored solutions satisfy all the constraints. All particles keep only those feasible solutions in their memory. Eleven test cases were tested and showed that PSO is an efficient and general solution to solve most nonlinear optimization problems with nonlinear inequality constraints.

Keywords: Particle swarm, optimization technique, nonlinear programming, evolutionary computation, nonlinear programming, constrained optimization

1. INTRODUCTION

Particle swarm optimization (PSO) is an evolutionary computation technique developed by Kennedy and Eberhart [1, 2]. It uses common evolutionary computation techniques: 1. It is initialized with a population of random solutions, 2. It searchs for the optimum by updating generations, and 3. Population evolution is based on the previous generations. In PSO, the potential solutions, called particles, are "flown" through the problem space by following the current optimal particles.

The update of the particles is accomplished by the following equations. Eq. (1) calculates a new velocity for each particle (potential solution) based on its previous velocity (V_{id}), the particle's location at which the best fitness so far has been achieved (p_{id} , or *pBest*), and the population global (or local neighborhood, in the local version of the algorithm) location (p_{gd} , *gBest* for global version or p_{ld} , *lBest* for local version) at which the best fitness so far has been achieved. Eq. (2) updates each particle's position in the solution hyperspace. The two random numbers c_1 and c_2 are independently generated. The use of the inertia weight w has provided improved performance in a number of applications [3].

$$V_{id} = w \times V_{id} + c_1 \times rand() \times (p_{id} - x_{id}) + c_2 \times Rand() \times (p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + V_{id} \quad (2)$$

Particle swarm optimization has been proven to be very effective for many optimization problems. Much work has been done in this area [4]. However, constrained optimization problems are still a new area for particle swarm optimization.

General constrained nonlinear optimization problems (CNOPs) can be defined as follows [5].

$$\text{Optimize } f(\vec{x}), \qquad \vec{x} = (x_1, x_2 \ldots x_n) \in \Re^N$$

$$\text{Where } g_j(\vec{x}) \le 0 \quad for \quad j = 1, \ldots, p \text{ and}$$

$$h_j(\vec{x}) = 0 \quad for \quad j = q+1, \ldots, m$$

They are made up of three basic components: a set of variables, a fitness function to be optimized (minimize or maximize) and a set of constraints that specify the feasible spaces of the variables. The goal is to find the values of the variables that optimize the fitness function while satisfying the constraints.

2. BACKGROUND

Over recent decades, significant research has been done on CNOPs. Due to the complexity and unpredictability of nonlinear optimization, a general deterministic solution is impossible. This provides an opportunity for evolutionary algorithms. In recent years, several evolutionary algorithms have been proposed for nonlinear optimization

problems. Michalewicz [6] provided an overview of these algorithms. In this paper, several evolutionary computation approaches were surveyed and a set of constrained numerical optimization test cases was provided.

The key point in the constrained optimization process is to deal with the constraints. Many methods were proposed for handling constraints. Koziel *et al.* [5] grouped them into four categories: methods based on preserving feasibility of solutions; methods based on penalty functions; methods that make a clear distinction between feasible and infeasible solutions; and other hybrid methods.

For CNOPs, the original PSO method needs to be modified to deal with constraints. Some ideas from the above constraint-handling methods can be adopted. The most straightforward one is the method based on preserving feasibility of solutions. In order to find the optimum in feasible space, each particle searches the whole space but only keeps tracking feasible solutions. And to accelerate this process, all the particles are initialized as feasible solutions. Following is the algorithm.

Figure 1: the Modified PSO Algorithm

```
For each particle {
    Do {
        Initialize particle
    } While particle is in the feasible space (i.e. it satisfies all the
    constraints)
}

Do {
    For each particle {
        Calculate fitness value
        If the fitness value is better than the best fitness value (pBest)
        in history AND the particle is in the feasible space, set current
        value as the new pBest
    }
    Choose the particle with the best fitness value of all the particles as
    the gBest
    For each particle {
        Calculate particle velocity according equation (a)
        Update particle position according equation (b)
    }
} While maximum iterations or minimum error criteria is not attained
```

There are two modifications compared to the original PSO.

1. During initialization, all the particles are repeatedly initialized until they satisfy all the constraints.
2. When calculating the *pBest* and *gBest* values, only those positions in feasible space are counted.

The above algorithm is the global version. For the local version, there is only one difference in the algorithm; instead of finding the *gBest*, each particle finds a neighborhood best (*lBest*) to update the new velocity.

3. EXPERIMENTAL DESIGN

Twelve constrained numerical optimization problems were tested in the experiment. They were proposed by Michalewicz and Schoenauer [6]. These test cases include objective functions of various types with different types of constraints [5, 6]. For detailed function information please refer to the references.

Table 1: 12 constrained nonlinear optimization test cases (From Koziel, *et al*, [5])

Func	Dim.	Type	Relative size of feasible space	LI	NE	NI
G1	13	Quadratic	0.0111%	9	0	0
G2	K	Nonlinear	99.8474%	0	0	2
G3	K	Polynomial	< 0.0001%	0	1	0
G4	5	Quadratic	52.1230%	0	0	6
G5	4	Cubic	< 0.0001%	2	3	0
G6	2	Cubic	0.0066%	0	0	2
G7	10	Quadratic	0.0003%	3	0	5
G8	2	Nonlinear	0.8250%	0	0	2
G9	7	Polynomial	0.5121%	0	0	4
G10	8	Linear	0.0010%	3	0	3
G11	2	Quadratic	< 0.0000%	0	1	0
G12	3	Quadratic	-	-	-	-

* LI: Linear inequalities, NE: Nonlinear equations, NI: Nonlinear inequalities

In Table 1, there are three test cases that deal with nonlinear equation constraints: G3, G5 and G11. In the PSO method, all randomly initialized solutions need to be located in the feasible space. So for those equation constraints, it is almost impossible to randomly generate a group of feasible solutions. There are several techniques to eliminate these equation constraints. One is to use direct replacement to remove the constraints, as follows:

- Test case G3a,
  Maximize $G3(\vec{x}) = (\sqrt{n+1})^{n+1} \cdot \prod_{i=1}^{n} x_i \cdot (1 - \sum_{i=1}^{n} x_i^2)$
  Subject to the following constraints
  $\sum_{i=1}^{n} x_i^2 \le 1$
  And bounds $0 \le x_i \le 1$, *for* $1 \le i \le n+1$
- Test case G11a,
  Minimize $G11(\vec{x}) = x_1^2 + (x_1^2 - 1)^2$
  Bounds $-1 \le x_i \le 1$, *for* $i = 1$

Test case G12 is a special case with 125 disjointed spheres of feasible space embedded in the search space.

In order to compare to others' results, two types of experiments were performed for each test case: Experiments #1: 20 runs were executed. For each run the maximum number of generations was set to T = 200. Experiments #2: 20 runs were executed. For each run the maximum number of generations was set to T = 500.

4. PARAMETER SELECTION

The population size of PSO is often between 10 and 40. Carlisle [7] compared different population sizes, and suggested 30 is a proper choice. Here, a population size of 20 was used. The reason for a lower population size is that it significantly lowers the computing time. This is because during initilization, all the particles must be in the feasible space. Randomly initialized particles are not always in the feasible space. So initilization may take a longer time if the population size is too large. However, for some complex cases, a larger population size is preferred.

In PSO, there are not many parameters that need to be tuned. Only the following several parameters need to be taken care of: maximum velocity V_{MAX}, inertia weight w, cognition learning rate c_1 and social learning rate c_2. Parameters settings were used as before [8; 9]. The inertia weight was [0.5 + (Rnd/2.0)]. The learning rates were 1.49445. The maximum velocity V_{MAX} was set to the dynamic range of the particle.

5. RESULTS

The following tables summarize the results of eleven test cases with nonlinear inequities constraints. All of the problems are changed to minimum optimizations. Thus all the optimum values showed here are minimum values.

First, it is important to note that PSO can successfully find the optimum for all the cases, except for the test case G5, which wasn't solved due to the equation constraints.

Table 2: Summary of the PSO results on experiments #1
(Global, Pop size 20, 200 iterations, 20 runs)
The test case G3 was run with k = 10 variables

Func.	Optimum	Worst	Best	Aver.
G1*	-15	-14.9579	-14.9987	-14.9880
G3a	-1.0	-0.99999	-1.0	-1.0
G4	-30665.5	-30665.4	-30665.5	-30665.5
G5	5126.4981	-	-	-
G6	-6961.8	-6821.1	-6943.5	-6899.5
G7**	24.306	31.9468	24.6269	26.6590
G8	-0.09583	-0.09583	-0.09583	-0.09583
G9	680.63	686.657	680.837	681.527
G10**	7049.33	8020.15	7219.25	7558.07
G11a	0.75	0.75000	0.75000	0.75000
G12	1.0	1.0	1.0	1.0

* Local PSO algorithm was used and population size was 50.
** Maximum iterations were changed to 2000.

Table 2 shows that for ten of the remaining eleven test cases, including two modified cases, G3a and G11a, PSO successfully found the optimum or near optimum in 200 iterations. Table 3 shows the results of the second experiment. Compared to other results, this is much faster than others have previously reported [5, 10].

Table 3: Summary of the PSO results on experiments #2
(Global, Pop size 20, 500 iterations, 20 runs)
The test case G3 was run with k = 10 variables

Func.	Optimum	Worst	Best	Aver.
G1*	-15	-15.0	-15.0	-15.0
G3a	-1.0	-1.0	-1.0	-1.0
G4	-30665.5	-30665.5	-30665.5	-30665.5
G5	5126.4981	-	-	-
G6	-6961.8	-6956.8	-6961.7	-6960.7
G7**	24.306	31.1843	24.4420	26.7188
G8	-0.09583	-0.09583	-0.09583	-0.09583
G9	680.63	681.675	680.657	680.876
G10**	7049.33	8823.56	7131.01	7594.65
G11a	0.75	0.75000	0.75000	0.75000
G12	1.0	1.0	1.0	1.0

* Local PSO algorithm was used and population size was 50.
** Maximum iterations were changed to 5000.

Besides its speed, PSO also found better results than those reported. Table 4 is a line-by-line comparison of the best test results from 20 runs. It shows that PSO got better or similar results with a much fewer iterations.

Table 4: Comparison of the best results achieved on ten test cases

Func.	Optimum	PSO 500*	GA 20000**
G1	-15	-15.0	-14.7864
G3a	-1.0	-1.0	-0.9997
G4	-30665.5	-30665.5	-30664.5
G5	5126.4981	-	-
G6	-6961.8	-6961.7	-6952.1
G7	24.306	24.4420****	24.620
G8	-0.095825	-0.0958250	-0.0958250
G9	680.63	680.657	680.91
G10	7049.33	7131.01****	7147.9
G11a	0.75	0.75	0.75
G12	1.0	1.0	1.0***

* Results are from table 2, 500 iterations
** Results are from Koziel *et al*., [5], 20000 iterations is used.
*** For G12, GA uses 500 iterations.
**** Maximum iterations were changed to 5000.

The global PSO algorithm was used except for test case G1. In the G1 case, the global PSO algorithm was sometimes trapped into a local minimum. This problem can be solved by using the local version of PSO algorithm. The population size was also changed to 50. PSO then found the global optimum without exception. For the test cases G7 and G10, a higher maximum iteration number was used. Maximum iterations were changed to 2000 and 5000 respectively. PSO also still got better results than those reported, and in a shorter time [5].

Test case G2 is the only case for which PSO did not locate the optimum area with the above settings. Different population size and maximum iteration numbers and local version of PSO algorithm were tried. The local PSO algorithm with a larger population size successfully found the global optimum. The results are shown in Table 5.

Table 5: Results under different settings for Test case G2
(Optimum value -0.8936)

Pop size	Local/ global	Max iterations	Result		
			Worst	Best	Aver
20	Global	5000	-0.3586	-0.7130	-0.5486
		10000	-0.3459	-0.7473	-0.5668
		20000	-0.2904	-0.7292	-0.5430
50	Global	10000	-0.3458	-0.7808	-0.5236
		20000	-0.4269	-0.7704	-0.5792
	Local	10000	-0.6316	-0.8033	-0.7521
		20000	-0.6437	-0.8036	-0.7643
		50000	-0.7778	-0.8036	-0.7957
200	Local	5000	-0.7138	-0.8027	-0.7660
		20000	-0.7935	-0.8036	-0.8023
		50000	-0.8023	-0.8036	-0.8035
500	Local	5000	-0.7550	-0.8028	-0.7871
		20000	-0.8005	-0.8036	-0.8034
		50000	-0.8036	-0.8036	-0.8036

6. DISCUSSION

It is known that different constraint-handling techniques provide different quality results for different CNOP cases. From the above results, we see that the PSO algorithm is consistent in locating the area of the global optimum in all CNOP cases. Due to the random origin of evolutionary algorithms, it is difficult to deal with the equation constraints. It is almost impossible to find a group of initial solutions in the feasible space. This also applies to those problems with extremely small feasible space.

Compared with other methods, PSO has the following advantages:

1. The algorithm is simple, there are not many parameters to be adjusted.
2. The algorithm is powerful, PSO is much faster for above benchmark functions, and the above results also show that it can deal with many kinds of optimization problems with constraints.
3. There is no predefined limit to the objective and constraints; it does not need to preprocess the objective and the constraints.

Two versions of PSO can be used in real problems. The local PSO algorithm is preferred for more accurate results while the global version is little bit faster. Higher iteration number and higher population size can be tried if the result is not satisfactory.

This paper presented a particle swarm optimization algorithm for constrained optimization. It demonstrated that PSO is an efficient and general method to solve most constrained parameter optimization problems. This paper represents only the first step in the investigation of solving constrained parameter optimization problems using particle swarm optimization. To be useful in practical applications, the ability of particle swarms to solve more complex constrained optimization problems will need to be proven.

REFERENCES

[1] Eberhart, R. C. and Kennedy, J. "A new optimizer using particle swarm theory", Proceedings of the Sixth International Symposium on Micro Machine and Human Science. Nagoya, Japan, pp. 39-43, 1995.

[2] Kennedy, J. and Eberhart, R. C. "Particle swarm optimization", Proceedings of IEEE International Conference on Neural Networks. Piscataway, NJ, IEEE service center. pp. 1942-1948, 1995.

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

[4] Kennedy, J., Eberhart, R. C., and Shi, Y., Swarm intelligence, San Francisco: Morgan Kaufmann Publishers, 2001.

[5] Koziel, S. and Michalewicz, Z., "Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization", *Evolutionary Computation*, vol. 7, no. 1. pp. 19-44, 1999.

[6] Michalewicz, Z. and Schoenauer, M., "Evolutionary Algorithms for Constrained Parameter Optimization Problems," *Evolutionary Computation*, vol. 4, no. 1, pp. 1-32, 1996.

[7] Carlisle, A. and Dozier, G. "An off-the-shelf PSO", Proceedings of the workshop on particle swarm optimization. Indianapolis, IN. Purdue School of Engineering and Technology, 2001.

[8] Hu, X. and Eberhart, R. C. "Tracking dynamic systems with PSO: where's the cheese?", Proceedings of the workshop on particle swarm optimization. Indianapolis, IN. Purdue School of Engineering and Technology, 2001.

[9] Eberhart, R. C. and Shi, Y. "Tracking and optimizing dynamic systems with particle swarms," Proceedings of the IEEE International Congress on Evolutionary Computation 2001. Seoul, Korea. pp. 94-97. 2001.

[10] Ray, T. and Liew, K. M. "A Swarm with an Effective Information Sharing Mechanism for Unconstrained and Constrained Single Objective Optimization Problem", Proceedings of IEEE International Congress on Evolutionary Computation, Seoul, Korea. pp.75-80, 2001.