International Journal of Students' Research in Technology & Management
eISSN: 2321-2543, Vol 6, No 2, 2018, pp 01-06
https://doi.org/10.18510/ijsrtm.2018.621

ANALYSING THE IMPACT OF PENALTY CONSTANT ON PENALTY FUNCTION THROUGH PARTICE SWARM OPTIMIZATION

[1]Raju Prajapati, [2]Om Prakash Dubey
[1]Amity University Jharkhand, Ranchi, Jharkhand - 834001 (India)
[2]Veer Kunwar Singh University, Ara, Bihar – 802301 (India)
Email: raju.prajapati20102011@gmail.com, omprakashdubeymaths@gmail.com

Article History: Submitted on 2nd January, Revised on 10th January, Published on 1st March 2018

Abstract. Non Linear Programming Problems (NLPP) are tedious to solve as compared to Linear Programming Problem (LPP). The present paper is an attempt to analyze the impact of penalty constant over the penalty function, which is used to solve the NLPP with inequality constraint(s). The improved version of famous meta heuristic Particle Swarm Optimization (PSO) is used for this purpose. The scilab programming language is used for computational purpose. The impact of penalty constant is studied by considering five test problems. Different values of penalty constant are taken to prepare the unconstraint NLPP from the given constraint NLPP with inequality constraint. These different unconstraint NLPP is then solved by improved PSO, and the superior one is noted. It has been shown that, In all the five cases, the superior one is due to the higher penalty constant. The computational results for performance are shown in the respective sections.

Keywords. Non-Linear Programming Problem, Penalty Method, Particle Swarm Optimization, Equality and Inequality Constraints.

INTRODUCTION

Non Linear Programming Problems are of two types, without constraints and with constraints. The NLPP without constraints could be modeled with several available methods like steepest descent, newton's method etc [1]. They are consuming comparatively less time as compared to the NLPP with constrains. The NLPP with constraint, on the other hand are modeled by various available methods like KKT conditions, Lagrangian multiplier, penalty method, barrier method etc [2, 3]. Almost all the various techniques available for NLPP with constrains are using one unique process: to convert NLPP with constraints to NLPP without constraints by imposing some constant, which we find during the further investigation directed in algorithm. After converting, these could be modeled by applying algorithms available for NLPP without constraints like steepest descent, newton's method etc.

There are some heuristic algorithm also, which could deal the NLPP without constrains. Particle Swarm Optimization is one of them. This meta heuristic was given by Kenndy and Eberheart in 1995 [4]. Till then, a lot of improvements over this has been made in a number of papers [5, 6]. A meta heuristic is an empirical strategy to find the solution in an alternate way. PSO works in a similar manner. The brief idea of PSO will be stated in the next section. We have used an improved version of this meta heuristic in the present paper to deal with the NLPP without constrains.

The NLPP with constraints are converted to NLPP without constraint by using penalty method. A penalty constant is needed for this purpose. Two different penalty constants are used for this purpose. Then, we apply improved PSO over both unconstrained NLPP to find the optimal solution. SCILAB programming language is used for the computational work involved. It has been shown that the higher penalty constant results sharper results. It is due to the higher convexity of the converted unconstrained NLPP.

The computational time taken by the improved PSO in this paper is lesser comparatively because we are generating only one random number in each iteration. The other random number is taken in such a way that the sums of these two are always one [7]. This also results a benefit on overall number of iterations, because we do not stuck with global or local impact only for the next iteration. Some more conventional improvements in PSO are also taken, which is discussed in the next section.

The PSO with penalty method is discussed in several papers [8]. We have compared our computational work with a similar work in [9], which is using 100 particles and 1000 iterations for an accuracy varying in order of 10^{-2} to10^{-3} i. e. the difference between the optimal function values and obtained values are varying between 10^{-2} and 10^{-3} for most of the problems. We have found the better accuracy (approximately in 10^{-3}) by taking 50 particles and 800 iterations. This is due to two main factors: i) higher penalty constant used for making more convex region and lesser search area, ii) taking improved version of PSO algorithm.

We apply the methods over the inequality constrained problems only. We intentionally took two variable problems as benchmark problems just because we could visualize the solution in 3D space. Also, the solution could be easily compared with the manual solution.

PARTICLE SWARM OPTIMIZATION

This section introduces briefly about PSO. The optimization algorithm PSO is, in general inspired by nature. Just for example, a flock of birds searches food on a given area, and as soon as some bird reaches to the food, it informs other. In this way, the search space got smaller and all the birds are accumulated on a particular area. The meta heuristic PSO works on similar grounds. We first generate the given set of solutions randomly, called the initial solution. The initial solutions differ from each others in "fitness". A solution may have a highest fitness, while the others may have lesser. The term highest fitness corresponds to the best solution obtained. Every other solution in the solution set must follow this best solution in a random way.

The best value of any particle achieved is called global best solution or 'gbest'. Each solution in the solution set is associated with a value, which is variable. These are called 'pbest'. At the initialization, gbest is the best position while pbest are particle's own position. In any iteraton, each particle is forced to move towards the best solution keeping track of also its own best position. The equation for movement of particle is given by:

$$v_{id}^{\ k+1} = v_{id}^{\ k} + c_1 r_1^{\ k}(pbest_{id}^{\ k} - x_{id}^{\ k}) + c_2 r_2^{\ k}(gbest_{id}^{\ k} - x_{id}^{\ k}) \quad \ldots (1)$$

Where $v_{id}^{\ k}$ denotes the velocity of i th particle at k th iteration. $r_1^{\ k}$ and $r_2^{\ k}$ are the two random numbers generated for each particle at the k th iteration. $pbest_{id}^{\ k}$ is the particle's own best position in ith iteration. $gbest_{id}^{\ k}$ is the best position obtained so far upto ith iteration. That means a particle is subjected to a velocity influenced by gbest and its own best position respectively. This velocity is also affected by two random numbers such that it may explore new position which is closer to its own best position pbest and global best solution gbest. The velocity obtained is added to its previous position for getting the new position.

$$x_{id}^{\ k+1} = x_{id}^{\ k} + v_{id}^{\ k+1} \quad \ldots (2)$$

The above processes explore and drift each particle towards the optimal solution. The process is stopped either when the desired accuracy is obtained or a specific number of iterations are completed.

IMPROVEMENT IN PSO

i) Inertia weight: The velocity equation of PSO could be modified as follow $v_{id}^{k+1} = wv_{id}^{k} + c_1 r_1^{k}(pbest_{id}^{k} - x_{id}^{k}) + c_2 r_2^{k}(gbest_{id}^{k} - x_{id}^{k})$, where the new constant w is called 'inertia weight' [5]. It alters the impact of previous velocity on the next velocity. In general, it's taken to be 1, but it has been found experimentally that the inertia weight value between 0.9 and 1.2 results a better improvement. Inertia weight affects the searh ability of particles. Larger the inertia weight, larger will be the global search ability. Similarly, smaller inertia weight results local search ability.

ii) Constriction factor: It was given by Eberhert and Shi in 2000 [5]. The velocity equation could also be modified as $v_{id}^{k+1} = \chi(v_{id}^{k} + c_1 r_1^{k}(pbest_{id}^{k} - x_{id}^{k}) + c_2 r_2^{k}(gbest_{id}^{k} - x_{id}^{k}))$, where χ is called the constriction factor. The value of χ is $\chi = \frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|}$, $\varphi = c_1 + c_2 > 4$, then the convergence speed is much faster. In general φ is taken to be 4.1. The constriction factor χ is then 0.729. The PSO with constriction factor is a special case of PSO with weight factor.

iii) Proper selection of particles: If we select all the initial particles closer to the optimal value, it would be easier to find the solution in a very less number of iteration. This is because of lower search space. Choosing a larger number of particles also give the solution in less number of iteration. This is because of increased probability of finding the position of optimal solution. The compound PSO was given by Angeline in 1998 [6], which gives an idea of basic selection strategy. For instance, a set of best particles could be chosen only for the next iteration.

iv) Proper random number generator: Consider the following velocity update equation [7] $v_{id}^{k+1} = v_{id}^{k} + c_1 r_1^{k}(pbest_{id}^{k} - x_{id}^{k}) + c_2(1-r_1^{k})(gbest_{id}^{k} - x_{id}^{k})$. This equation is using only one random number r_1, reducing the computational time for generating two random number. If there were two random numbers r_1^{k} and r_2^{k} and both are closer to zero, we get almost no impact of pbest or gbest on v_{id}^{k+1}. But if we take r_2^{k} as 1-r_1^{k}, we have an impact on v_{id}^{k+1}.

All the improvements mentioned above are incorporated in our present paper. For instance, we are choosing initial particle selection strategy in a smaller region. We are imposing a combined constriction factor and weight factor also, which results an update of w, c_1, and c_2. We are also choosing one random number for any iteration.

PENALTY METHODS

This section introduces briefly about the penalty method we are about to use. We are defining this method in the context of solving non-linear programming problem (NLPP) with inequality constraint (as we are concern only with inequality constraint problems here). Let's consider the NLPP with n constraint as:

$$\begin{aligned} &\text{Minimize} && f(x) \\ &\text{Subject to} && g_i(x) \le 0, i = 1,2....m \end{aligned} \quad \dots (3)$$

The above NLPP with constraint could be solved by converting it into NLPP without constraint by a method called 'penalty method'. If we apply the penalty method for an inequality constraint [10], the penalty will be,

$$\max(0, g_i(x))^2 \text{, for constraint i.} \qquad \dots (4)$$

That means if the constraint is violated, a penalty must be imposed for that. If the constraint is not violated, the penalty becomes 0 automatically from the above formula. The index 2 is used for making the value of imposed penalty positive. This penalty must be added or subtracted to the minimization or maximization problems respectively.

An amplifying constant, called penalty constant is used for making the penalty more sharper/higher. Therefore the penalty function becomes

$$\text{c.} \max(0, g_i(x))^2 \text{, for constraint i.}$$

Also, the unconstraint problem becomes

$$\text{Minimize } f(x) + c.\max(0, g_i(x))^2 . \qquad \dots (5)$$

ROLL OF CONSTANT IN PENALTY FUNCTION

The constant is used for making the penalty large. The convexity of minimization problem above depends on the constant c. If the value of c is higher, convexity is higher. More convex region means, lesser search space could be achieved easily. It would be better to have much larger value of constant c. But we can't take much higher values due to constraint by software.

METHODOLOGY

We have applied the penalty method over NLPP with constraints to convert into NLPP without constraints. Then we are solving the NLPP without constraints by improved PSO methods. The improved version of PSO means, we are taking care of each factor. These factors are already discussed above in earlier sections.

We are more concern with the penalty constant c in this paper. For that reason, we are taking two values of c, they are 10 and 100. Then SCILAB programming language is used to implement the PSO over NLPP without constraints. We are taking five test problems for this purpose. All are solved first by taking c=10, then by taking c=100 respectively. The five test problems are intentionally taken to be easier and with two dimensional only, so that the solution could be visualized/obtained manually also. Then, It would be easier to compare with the computationally obtained solutions. Also, the graph of the function could be drawn in a three dimensional space.

The following five test problems are used:

Test problem 1: Minimize $x + y$, subjected to $x^2 + y^2 \leq 1$. Manual solution is $-\sqrt{2}$ at $(x, y) = (-1/\sqrt{2}, -1/\sqrt{2})$.

Test problem 2: Minimize $(x-10)^2 + (y-10)^2$, subjected to $x + y \geq 20 + \sqrt{2}$. Manual solution is 1 at $(x, y) = (10 + 1/\sqrt{2}, 10 + 1/\sqrt{2})$.

Test problem 3: Minimize $-xy$, subjected to $x+y \le 1, x+y \ge 1$. Manual solution is -0.25 at $(x,y)=(1/2,1/2)$.

Test problem 4: Minimize x^2+y^2, subject to $x+y \ge 1, y \le x-1$. Manual solution is 1 at $(x,y)=(1,0)$.

Test problem 5: Minimize $x+y-2$, subjected to $y-1 \ge (x-1)^2$. Manual solution is -.25 at $(x,y)=(1/2,5/4)$.

COMPUTITIONAL RESULTS

We have conducted the experiments with 50 initial particles. Two values of penalty constants (c=10 and c=100) are selected. Over 800 iterations are conducted in each experiment in SCILAB. The weight factor is taken to be negligible (0.01), constriction factor as 1. The search region is taken approximately as 10 units. The values of each experiment are noted, and compared with the manual values.

CASE 1: c=10

Test function	x value	y value	Optimal value
1	-.7202833	-.7182556	-1.4264986
2	10.673949	10.672712	.9523822
3	.5108446	.5147917	-.2564064
4	.9090909	0	.9090909
5	.4949041	1.2055173	-.2749725

CASE 2: c=100

Test function	x value	Y value	Optimal value
1	-.7144323	-.7017164	-1.4153538
2	10.689868	10.715096	.9958356
3	.4879969	.5145776	-.2504495
4	.9900990	0	.9900990
5	.4830205	1.2622229	-.2522115

CONCLUSIONS

We have conducted the computational experiment successfully and found that the results are sharper when higher convexity is used. The results in terms of optimal value are close to the manual values in order of 10^{-3} approximately. When compared with the results in [9], we have found an improvement with apparently lesser number of particles and lesser number of iterations. This is due to two main things used in the experiments. They are i) Improvement in choosing the random number during PSO experiments and ii) Choosing higher value of penalty constant. We have applied the test over smaller problems. Further, this work could be extended to some well-known test functions like Rastrigin function, Rosenbrook function etc. The research could also be extended to the equality constraints, in which Penalty and Barrier methods behaves alike.

REFERENCES

[1] Battiti, R., First-and second-order methods for learning between steepest descent and Newton's method. Neural computation, 1992, 4(2), pp.141-166.

[2] Fiacco, A.V. and McCormick, G.P., Nonlinear programming: sequential unconstrained minimization techniques. Society for Industrial and Applied Mathematics, 1990.

[3] Kelley, C.T., Iterative methods for optimization. Society for Industrial and Applied Mathematics, 1999.

[4] Eberhart, R. and Kennedy, J., A new optimizer using particle swarm theory. Micro Machine and Human Science, 1995. MHS'95, Proceedings of the Sixth International Symposium, pp. 39-43. IEEE.

[5] Eberhart, R.C. and Shi, Y., Comparing inertia weights and constriction factors in particle swarm optimization. Evolutionary Computation, 2000. Proceedings of the 2000 Congress, Vol. 1, pp. 84-88. IEEE.

[6] Angeline, P.J., Using selection to improve particle swarm optimization. Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence, The 1998 IEEE International Conference pp. 84-89.

[7] Li, W.T., Shi, X.W. and Hei, Y.Q., An improved particle swarm optimization algorithm for pattern synthesis of phased arrays. Progress In Electromagnetics Research, 2008, 82, pp.319-332.

[8] Vardhan, L.A. and Vasan, A., Evaluation of penalty function methods for constrained optimization using particle swarm optimization. Image Information Processing (ICIIP), 2013 IEEE Second International Conference, pp. 487-492.

[9] Parsopoulos, K.E. and Vrahatis, M.N., Particle swarm optimization method for constrained optimization problems. Intelligent Technologies–Theory and Application: New Trends in Intelligent Technologies, 2002, 76(1), pp.214-220.

[10]https://web.stanford.edu/group/sisl/k12/optimization/MO-unit5-pdfs/5.6penaltyfunctions.pdf (Retrieved: 28th October 2017)