International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 4 (2015) pp. 9101-9110

http://www.ripublication.com

Accurate Solution of Benchmark Linear Programming Problems Using Hybrid Particle Swarm Optimization (PSO) Algorithms

***Geraldine Bessie Amali.D[1] and Vijayarajan. V[2]**
School of Computing Science and Engineering, VIT University
Vellore– 632014, Tamilnadu, India
[1]geraldine.amali@vit.ac.in*[2]vijayarajan.v@vit.ac.in
Phone :[1]9489134212*[2]9841101779

Abstract

Solution of large scale linear programming problems is of fundamental importance in optimal resource allocation. The Simplex Method that is primarily used to solve linear programming problems computes the optimal solution in finitely many steps. However the number of steps required by the Simplex Method is not fixed and can be very large since the number of corners of the feasible set that will be visited cannot be predicted in advance. Hence the worst case complexity is exponential. Biologically inspired Particle Swarm Optimization (PSO) algorithm has been successfully used to solve a wide variety of difficult optimization problems and demonstrated to outperform other algorithms like Genetic Algorithms. In this paper hybrid PSO algorithms like Nelder-Mead Particle Swarm Optimization (NMPSO) and Gradient Particle Swarm Optimization(GPSO) have been applied to solve Linear Programming Problems. A new projection operator is introduced in this paper to project vectors with negative components into the feasible set since solutions with negative components are not feasible solutions. Performance of the hybrid PSOs and classical PSO are compared using benchmark linear programming problems.

Keywords: Particle Swarm Optimization, Nelder-Mead PSO, Gradient Particle Swarm Optimization algorithm

Introduction

Particle Swarm Optimization Algorithm [6] has been inspired by the social behavior of birds. The algorithm is simple and performs better than other optimization algorithms like genetic algorithms [6]. For the linear programming problem in the form:

$$minimize\ c_1x_1 + c_2x_2 + \cdots + c_nx_n$$
$$\text{subject to } Ax = b \text{ and } x_i \geq 0 \quad (1)$$

Our objective is to minimize the cost function cx which is subject to constraints. The PSO algorithm starts with a population of points (also called particles) which are randomly generated in the search space. For every particle generated in the search space, best solution encountered so far is computed. The PSO algorithm moves each particle towards the best solution found by the particle (pbest) and found by the entire population (gbest). Local search around the global best solution is done by Nelder-Mead method[7]. Nelder-Mead method is a local search algorithm. Another local search algorithm that we use is the gradient particle swarm optimization algorithm(GPSO)[3]. Gradient Particle Swarm Optimization algorithm uses the particle swarm optimization algorithm to find the global solution and a gradient descent scheme for accurate local exploration.

Particle Swarm Optimization Algorithm

This algorithm is a population based stochastic optimization algorithm. In classical PSO algorithm each particle is moved toward a random average of the best position found by that particle and the best position found by the entire set of particles. The update formula for the PSO algorithm is given in equation (2).

$$V_{id}(k+1) = V_{id}(k) + \emptyset_{id}\left(P_{id}(k) - X_{id}(k)\right) + \emptyset_{2d}\left(G_d(k) - X_{id}(k)\right) \quad (2)$$
$$X_{id}(k+1) = X_{id}(k) + V_{id}(k)$$

where V_i= [$V_{i1}V_{i2} \ldots . V_{iN}$] is the velocity for particle i; X_i=[$X_{i1}X_{i2} \ldots . X_{iN}$] is the i^{th} particle's position; $\emptyset_{1d}$and $\emptyset_{2d}$ are random numbers which are distributed uniformly; P_i= [$P_{i1}P_{i2} \ldots . P_{iN}$] is the best position determined by particle i; $G = [G_1G_2 \ldots G_{n]}$ is the best position found by the entire set of particles; the search space dimension is denoted by N, and the iteration number is represented by k. Whenever a particle position is changed, the global best solution will be replaced by the new solution if $f(X_i) < f(G)$.

Penalty Function Method

The PSO, NMSPO and GPSO algorithms are unconstrained optimization algorithms and cannot as such be used to solve constrained optimization problems [4]. Here, we use the penalty function method [2] to transform the linear programming problem into an unconstrained optimization problem. In the penalty method instead of minimizing $f(x)$ subject to constraints the function $f(x) + \gamma P(x)$ (where γ is a positive constant) is minimized. The function $P(x)$ is referred to as the penalty function. The penalty function is a continuous nonnegative function that is zero in the feasible set and takes on large values for points far away from the boundary of the feasible set. When $f(x) + \gamma P(x)$ is minimized $f(x)$ and $P(x)$ are separately minimized. However the minimum value of $P(x)$ is zero so effectively $f(x)$ is minimized.

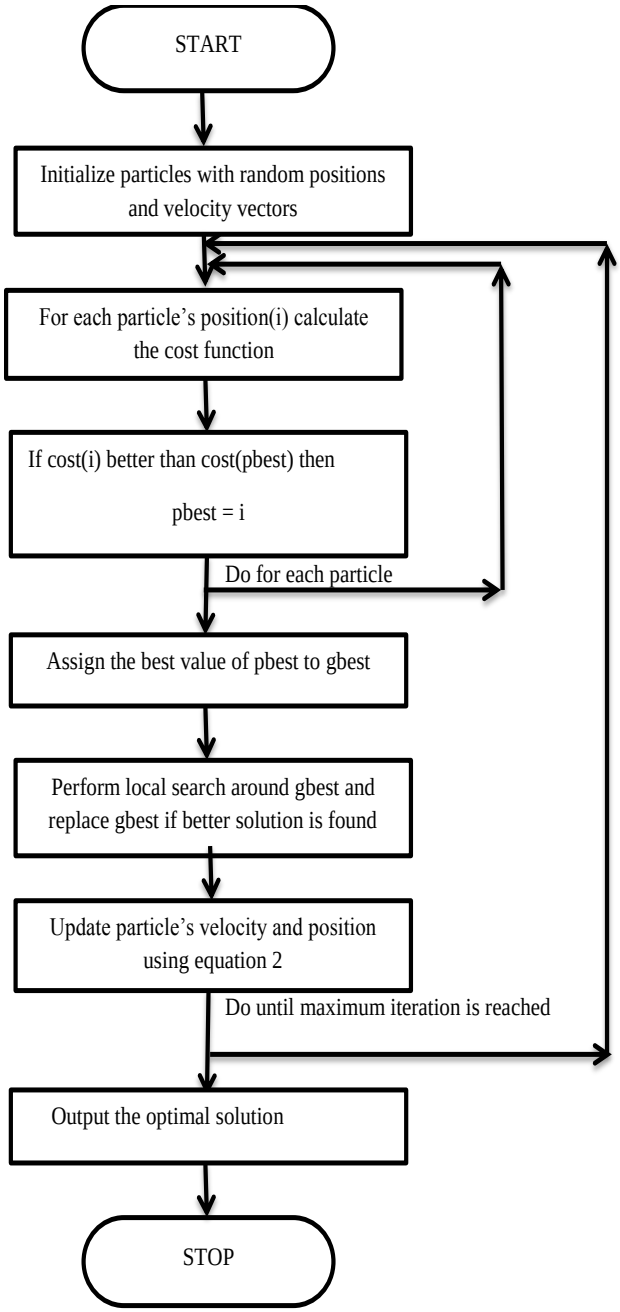

Figure 1: Flowchart depicts the working of the hybrid PSOs. In the case of NMPSO, Nelder Mead local search algorithm is used to perform the local search around the obtained best solution and a gradient descent scheme is used in the case of GPSO to improve the accuracy of the solution obtained by the PSO.

Proposed System

The following penalty function is proposed to convert the linear programming problem

$minimize\ c_1x_1 + c_2x_2 + \cdots + c_nx_n$subject to $Ax = b$ and $x_i \geq 0$ into an unconstrained optimization problem:

$$minimize\ c_1x_1 + c_2x_2 + \cdots + c_nx_n + \gamma_1\|Ax - b\|^2 + \gamma_2\|\min(x, 0)\|^2 \quad (3)$$

$\gamma_1$andγ_2 are chosen to be large positive constants to penalize violation of the constraints. From (3) it is clear that if during the search an intermediate solution violates the constraints a large positive penalty is added to the cost function so it will be rejected as a candidate for the minimum. In this paper the following new projection operator (4) is proposed to speedup convergence:

$$x \leftarrow \max(x, 0) \quad (4)$$

Where $x = [\, x_1\, x_2\, x_3\, \ldots\, x_N\,]$ is vector with N real components and the max() operation is done component wise. The use of this operator ensures that solutions with negative components are eliminated since vectors with negative components are not feasible. In this paper global search of the PSO algorithm is augmented with local search to increase the accuracy of the final solution. Nelder-Mead Simplex method and gradient based BFGS algorithms are used for local search to improve the accuracy of the solution computed by the PSO algorithm.

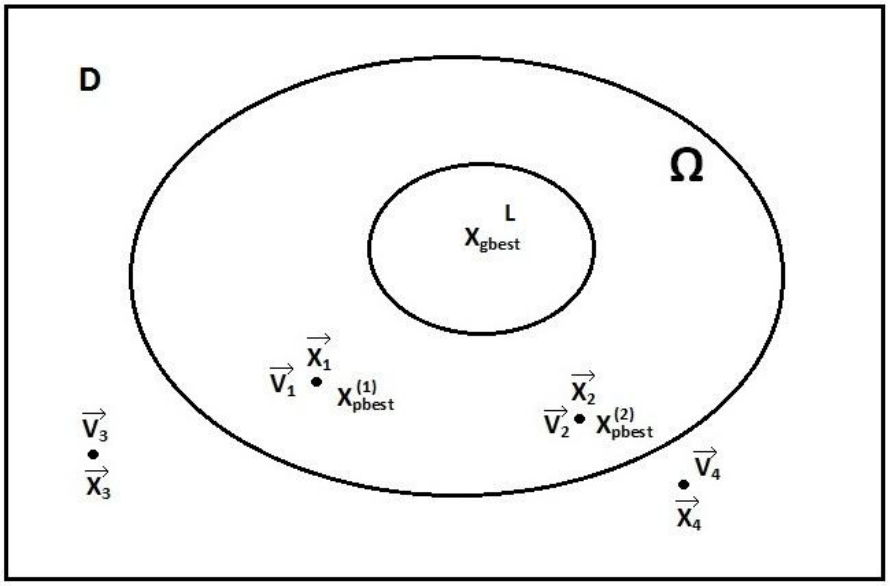

Figure 2: Figure illustrates the working of the PSO algorithm in the search space D and the feasible set Ω.$\overrightarrow{X_i}$and$\overrightarrow{V_i}$are the position and velocity vectors of the i[th] particle referred in equation(2). L is the set of points around X_{gbest} in which the local search is done by the hybrid PSOs.

Each particle is represented as a struct with the following attributes
Particle(k).x – Position of the k^{th} particle
Particle(k).v – Velocity of the k^{th} particle
Particle(k).pbest – Best solution found by the k^{th} particle
Particle(k).cost – Value of the cost function at the location of the k^{th} particle

Results and Discussion

Comparison of PSO, NMPSO and GPSO on benchmark linear programming problems

Benchmark linear programming problems [4] were taken and the three algorithms were implemented in a Mat lab environment. The best solution and the values of the variables were computed. A graph was plotted for each algorithm for 100 iterations .It was found that the hybrid PSO algorithms performed better than the classical PSO.

Since we know that negative numbers cannot be in the feasible set, GPSO works in such a way that every time a particle is updated negative elements are replaced by zero thereby reducing the search area which aids in faster convergence. Local search is done around the newly found solution. If the local search gives a better solution then the previous value is replaced by the newly found best solution.

The PSO update formula does not guarantee that solutions will have non negative components. In this paper projection operator is used to project vectors with negative components into the feasible set since solutions with negative components are not feasible solutions for a linear programming problem. Simulation results show that the use of the projection operator to guarantee positive components leads to faster convergence as shown in Fig.6.

The use of the projection operator leads to faster convergence than results reported in recent literature [4]. Convergence of the algorithms and the best solutions found is shown in Table 1 and Figures 3 to 9.

Problem 1

Objective function:

$$minimize \; -2.2361x_4 + 2x_5 + 4x_7 + 3.6180x_8 + 3.236x_9 + 3.6180x_{10} + 0.764x_{11}$$

Subject to:

$$x_1 = 1$$

$$x_2 + 0.3090x_4 - 0.6180x_5 - 0.8090x_6 - 0.3820x_7 + 0.8090x_8 + 0.3820x_9 + 0.3090x_{10}\, 0.6180x_{11} = 0$$

$$x_3 + 1.4635x_4 + 0.3090x_5 + 1.4635x_6 - 0.8090x_7 - 0.9045x_8 - 0.8090x_9 + 0.4635x_{10} + .3090x_{11} = 0$$

Problem 2
Objective function:
$minimize\ 2x_1 + 4x_4 + 4x_6$
Subject to:

$$x_1 - 3x_2 - x_3 - x_4 - x_5 + 6x_6 = 0$$
$$2x_2 + x_3 - 3x_4 - x_5 + 2x_6 = 0$$

Table 1:

Values	Best Solution (from Reference)	Best Solution (from PSO)	Best Solution (from NMPSO)	Best Solution (from GPSO)
Fmin	0	59.0646	8.5881 E-004	2.1583e-007
X1	1	1.0000	1.0000	1.0000
X2	0	0.9554	0.0000	0
X3	0	2.3544	0.0000	0
X4	0	1.4952	0.0000	0
X5	0	6.3246	0.0001	0
X6	0	0.3145	0.0000	0
X7	0	6.4033	0.0001	0
X8	0	3.8325	0.0000	0
X9	0	0.0717	0.0000	0
X10	0	2.3162	0.0000	0
X11	0	2.1814	0.0001	0

Note: The value zero in Table-1 refers to any value less than machine precision.
Best solution found for benchmark problem 1 over 100 independent runs is shown

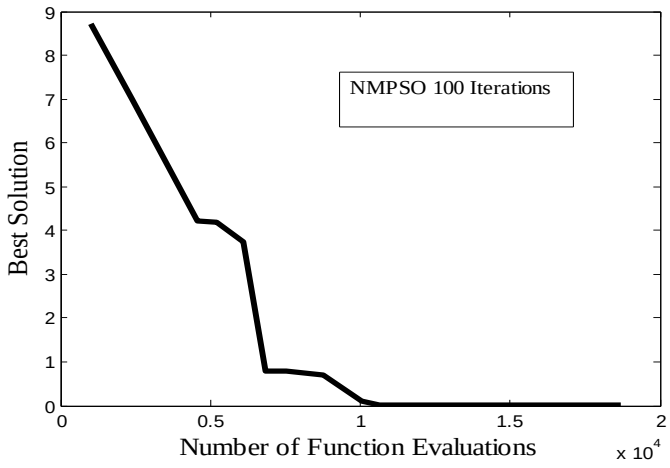

Figure 3: Convergence of the NMPSO algorithm for benchmark problem 1. Population size=20, number of iterations=100, search space= $[0,10]^{11}$

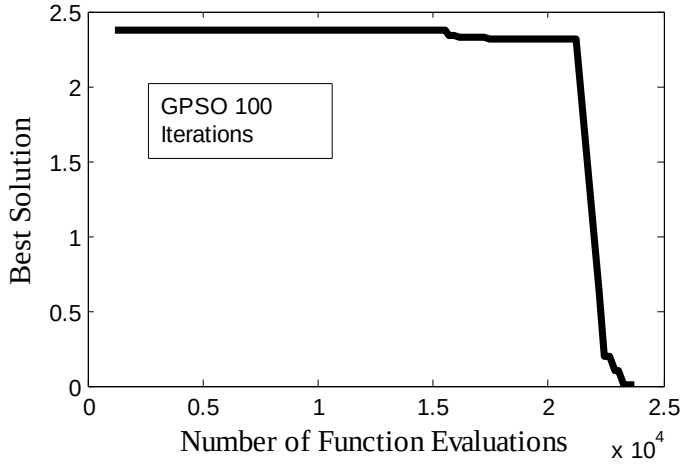

Figure 4: Convergence of the GPSO algorithm for benchmark problem 1. Population size=20, number of iterations=100, search space= $[0,10]^{11}$

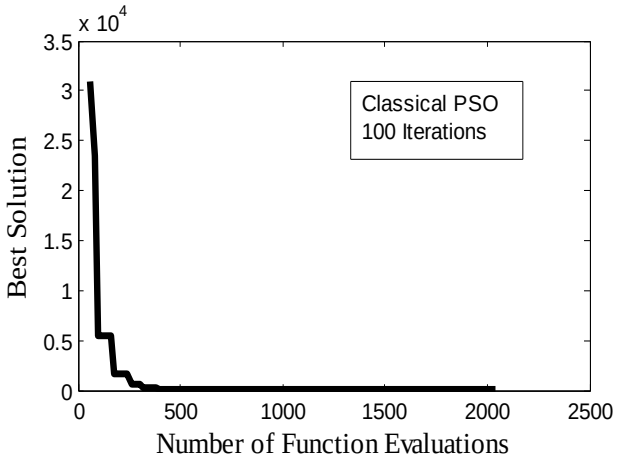

Figure 5: Convergence of the Classical PSO algorithm for benchmark problem 1. Population size=20, number of iterations=100, search space= $[0,10]^{11}$

Problem number 2 was executed over 100 independent runs in a Matlab programming environment with a floating point relative accuracy of 2^-52 which is 2.2204e-016. The algorithm converged to values less than machine precision and matched the best solution from reference [4].

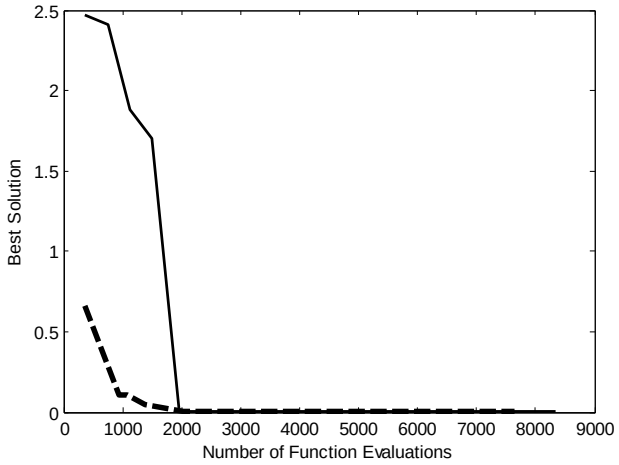

Figure 6: The dashed line shows that the use of the projection operator leads to faster convergence

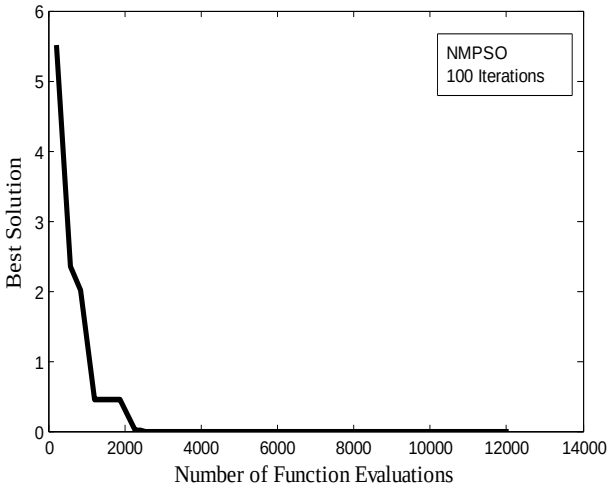

Figure 7: Convergence of the NMPSO algorithm for benchmark problem 2. Population size=20, number of iterations=100, search space= $[0,10]^{11}$

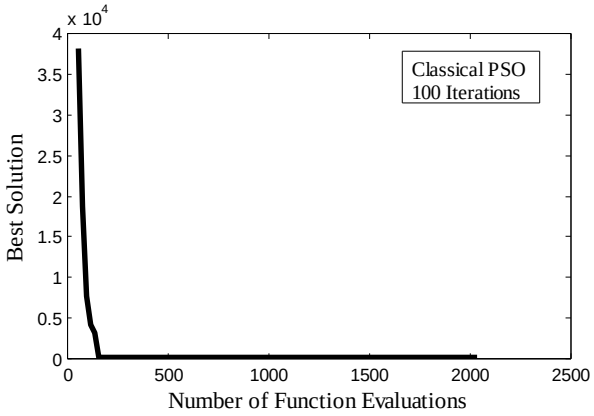

Figure 8: Convergence of the Classical PSO algorithm for benchmark problem 2. Population size=20, number of iterations=100, search space= $[0,10]^{11}$

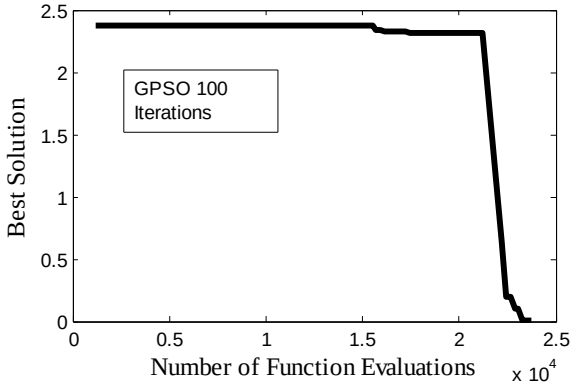

Figure 9: Convergence of the GPSO algorithm for benchmark problem 2. Population size=20, number of iterations=100, search space= $[0,10]^{11}$

Conclusion

Theclassical PSO, GPSO and NMPSO were implemented using the newly proposed projection operator along with the penalty function method that was tested with the benchmark problems. It was found that the hybrid PSO with new projection operator has performed better than classical PSO. The algorithms converged to better solutions than those found in the literature. The simulation results show that the use of the projection operator to guarantee positive components leads to faster convergence. As with any biologically inspired optimization algorithm our approach is computationally intensive. Dividing the population size into sub populations and later making them communicate with each other will greatly increase the performance. The performance of parallel implementations of the PSO algorithm on large scale linear programming problems will be considered in future work.

References

[1] Strang, G(1986), "Introduction to Applied Mathematics", Cambridge Press, Wellesley

[2] Chong, K.P.E., Zak, H.S.(2010), "An Introduction to Optimization", ChoudharyPress,Delhi

[3] Mathew M. Noel, "A new gradient based particle swarm optimization algorithm for accurate computation of global minimum", January 2012, Applied Soft Computing, Volume 12, Issue 1, Pages 353-359, ISSN 1568-4946, 10.1016/j.asoc.2011.08.037.

[4] PakizeErdogmus,"Particle swarm optimization performance on special linear programming problems", 18 June, 2010Scientific Research and Essays Vol. 5(12), pp. 1506-1518.

[5] Yan Jiang, Tiesong Hu, ChongChao Huang, Xianing Wu, "An improved particle swarm optimization algorithm", 1 October 2007.Applied Mathematics and Computation, Volume 193, Issue 1, Pages 231-239, ISSN 0096-3003, 10.1016/j.amc.2007.03.047.

[6] R.C.Eberhart and Y.Shi, "Comparison between Genetic Algorithms and Particle Swarm Optimization", Springer 1998, Evolutionary Programming VII, Lecture Notes in Computer Science 1447, pp.611-616.

[7] ErwieZahara, Yi-Tung Kao, "Hybrid Nelder Mead simplex search and particle swarm optimization for constrained engineering design problems", March 2009, Expert Systems with Applications. Volume 36, Issue 2, Part 2, Pages 3880-3886, ISSN 0957-4174,0.1016/j.eswa.2008.02.039.