Comparison of Optimization of Helical Compression Spring for Minimum Weight Using Genetic Algorithm and PSO

Arvind Gothwal
Department of Mechanical Engineering,
SPIT Piludara, Mehsana,Gujarat 384001, India

Abstract: Helical spring is very useful machine element and it is widely use in engineering industries such that suspension, engine valve, weighting machine etc. The present work is about comparison of optimization of helical compression spring using genetic algorithm and particle swarm optimization technique (PSO). This will be helpful to selection the technique for optimization problems in engineering, software, management etc. Optimization of helical spring for minimum weight reduce the metal requirement and it is beneficial for industries.

Index Terms **– Genetic Algorithm, Particle Swarm Optimization, Helical Compression Spring**

I. INTRODUCTION

The applications of helical spring are cam and follower, clutch, engage and disengage of transmission system, in governors, spring balance etc. Helical springs find their application in every aspect of mechanical engineering. So a lot of attention has been paid to its design optimization. Agrawal [1] studied the design of helical spring for minimum weight by geometric programming. Explicit solution of the optimization problem was obtained and was applied to a numerical example. Azarm et al. [2, 3] optimized the helical spring with the help of consol-optcad and extended his work to the optimization of flywheel. Ponsich et al [4] presented optimal batch plant design in Process Engineering, which can be formulated as a Mixed Integer Non-Linear Programming optimization problem involving specific constraints and determined the best fitted. Appala et al [5] used a non-traditional optimization technique namely Genetic Algorithm for Optimal design of flywheel. Many researchers worked on optimization problem using genetic algorithm and particle swarm optimization so this work for comparison is chosen here.

II. Problem formulation and results

A helical compression spring problem is taken and table 1 and table 2 are showing are showing objective function, constraints, material properties and limitations of different parameters of spring like spring wire diameter, coil diameter and number of turns.

Table 1 Objective function and constraints

Objective function and Constraints[7]
$w = 19.39 \times 10^{-6}(n+1)Dd^2$
$g_1 = 0.356\frac{d^3}{D} \times \frac{1}{(\frac{4D-d}{4D-4d}+\frac{0.615d}{D})} - 2$
$g_1 = \frac{10171.25d^4}{D^3 n} - 15 = 0$
$g_3 = \frac{0.1474x_2^3 x_3}{x_1^4} - 100 = 0$

Table 2 properties of material, load conditions and limitations

Input parameters for helical spring design	
Maximum force	1500N
Stiffness of spring	15N/mm
Ultimate tensile strength	$1360\ N/mm^2$
Modules of rigidity	$81370\ N/mm^2$
Density of material	$7.861 \times 10^{-6}\ kg/mm^3$
Factor of safety(f)	2
Diameter of wire(d)	5 to 9
Diameter of coil(D)	35 to 50
Number of turns(n)	15 to 25
Spring Index(C)	6 to 8

Formulated problem is solved by genetic algorithm and particle swarm optimization for comparison, first comes on genetic algorithm and then PSO.

Genetic algorithm:
Genetic algorithm is a tool which can optimize any equation with constraint or without constraints. Algorithm starts with encoding. Here a set of solution represented by chromosomes that is population. Population size is constant in all iteration. This chromosome generates new chromosomes with help different operators that are crossover and mutation. After using operators new population

comes and better chromosomes or set of solution will be select for next iteration. After number of iteration algorithm converges to best solution and it may be optimal or sub optimal. Genetic algorithm has four tool which help to optimization that are Encoding, Decoding, Selection, crossover and mutation.
Encoding is a key to generate a set of solutions for evolution. Mostly binary codes use for this operation.
If a function $[f(x_1, x_2, x_3, \dots, x_N)]$, then one chromosome represents a function cost.

101011|111011|100111|…….|101101

Fig.1: Representing chromosome in binary coding.[6]

In Figure 1, one chromosome has $(6 \times N)$ bits and 6 bits represents one variable and function depends on N variable. Accuracy of function depends on number of bits. Number of bits and number of chromosomes in population varies according to difficulty of problem.

Decoding is use to convert chromosomes, binary to decimal form in a required range.
Formula used for decoding given by,

$$x_i = x_i^L + \frac{x_i^U - x_i^L}{2^n - 1} \quad (1)$$

Where n is the no. of bits used per variable,
$x_i^L \rightarrow$ Lower limit of i^{th} variable,
$x_i^U \rightarrow$ Upper limit of i^{th} variable,
$x_i \rightarrow$ Decoded value of i^{th} variable

Selection is a process to find better chromosomes from population and how many times it should be selected. Chances of selection is depends on fitness value of that chromosome.

Crossover is an operator use for crossing two chromosomes and produces two new chromosomes.

101011|111011|100111|…….|101101 (P_1) **101**101|**11**0111|**100**101|…….|110010 **(C_1)**
001101|100111|110101|……..|110010 (P_2) 001**011**|10**1011**|110**11**|……..|1**01101** (C_2)

Fig. 2: Showing new chromosomes produced for evaluation.

Where C_1, C_2 are children produced by parents P_1, P_2

Mutation is an operator which adds new information at random way. Randomly selects new chromosomes and applies mutation. It removes local optima and it help to generates global minima.

Steps are for genetic algorithm

Step1: define objective function, bounds of variables, constraints and generate number of populations
$i \leq$ maxiteration $iteration$
Step2: decode all populations and find minimum value of objective function and set ${f_{min}}^i$ as, minimum value for i^{th} iteration
Step3: use selection, crossover and mutation for i^{th} iteration
Step4: decode all populations and find minimum value of objective function and set ${f'_{min}}^i$ as, minimum value for i^{th} iteration
Step5: if ${f'_{min}}^i < {f_{min}}^i$ than reset population $pop({f_{max}}^i) = pop({f'_{min}}^i)$ and i=i+1
Step6: else i=i+1 and repeat from step 3.

End

Table 2b Data selected for Genetic Algorithm

Number of populations	No. of bits in one chromosome	Type of coding	Type of crossover	Crossover probability	Probability of mutation	Type of selection	Number of Iterations
60	10	Binary	Single point crossover	0.8	0.06	Rolette Wheel	200

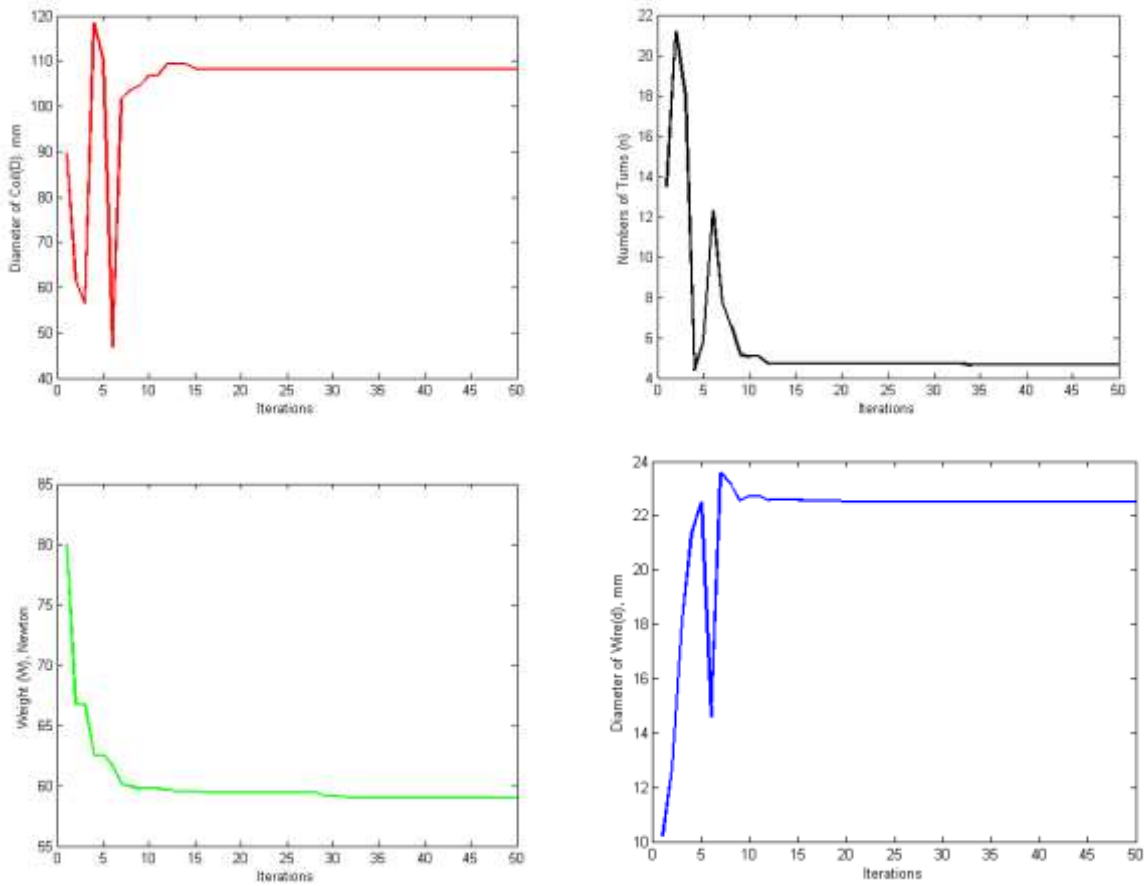

Fig 3 Optimized Results by Genetic Algorithm

Particle Swarm Optimization:

PSO algorithm is inspired by the behavior of biological organisms, specifically movement of groups of some species of animals in a given area, e.g. birds flocking to find a food source. This behavior is used in optimization search for solutions in a real-valued search space. PSO starts with number of solutions and each solution is called particle. Each particle has its velocity and a position. Each particle searches for best and updates its velocity and position. It searches for optimum so it updates velocity according to p_{best} *and* g_{best}. Where,p_{best} - is better position of particle between old and new. g_{best}, is the best position among all particles. In each iteration particle updates its position and velocity, and solution improves. Three main steps of PSO are

1. Evaluate the fitness of each particle
2. Update individual and global best finesses and positions
3. Update velocity and position of each particle

Updating velocity [8] and particle new position is obtained according to equation (2), (3) and (4).

$$V_{new} = w \times V_{old} + C_1 \times (rand) \times (p_{best} - p_{old}) + C_2 \times (rand) \times (g_{best} - p_{old}) \quad (2)$$

$$p_{new} = p_{old} + V_{new} \quad (3)$$

$$w = (t - i)/t \quad (4)$$

Where

w - Inertia weight constant
V_{new} - New velocity calculated for each particle
V_{old} - Velocity of the particle from the previous iteration
p_{new} - New position calculated for each particle
p_{old} - Position of the particle from the previous iteration
$C_1 \& C_2$ - Cognitive and social acceleration constants
$rand$ - Generates a random value in the range [0 1]
t - Total number of iteration
i - i^{th} iteration

C_1 and C_2 are generated in the range 0.4 to 2[9]
Initially large value of inertia weight is there to promote global exploration of search space. Inertia weight is decreased in accordance to eq.(3) to facilitate exploitation. The most common approach for solving constraint optimization is to use penalty function so that problem is transformed into unconstrained by penalizing the constraints.

Table 3 Specification of genetic algorithm

Number of populations	C_1	C_1	**Number of iteration** t
30	1.49	1.8	1000

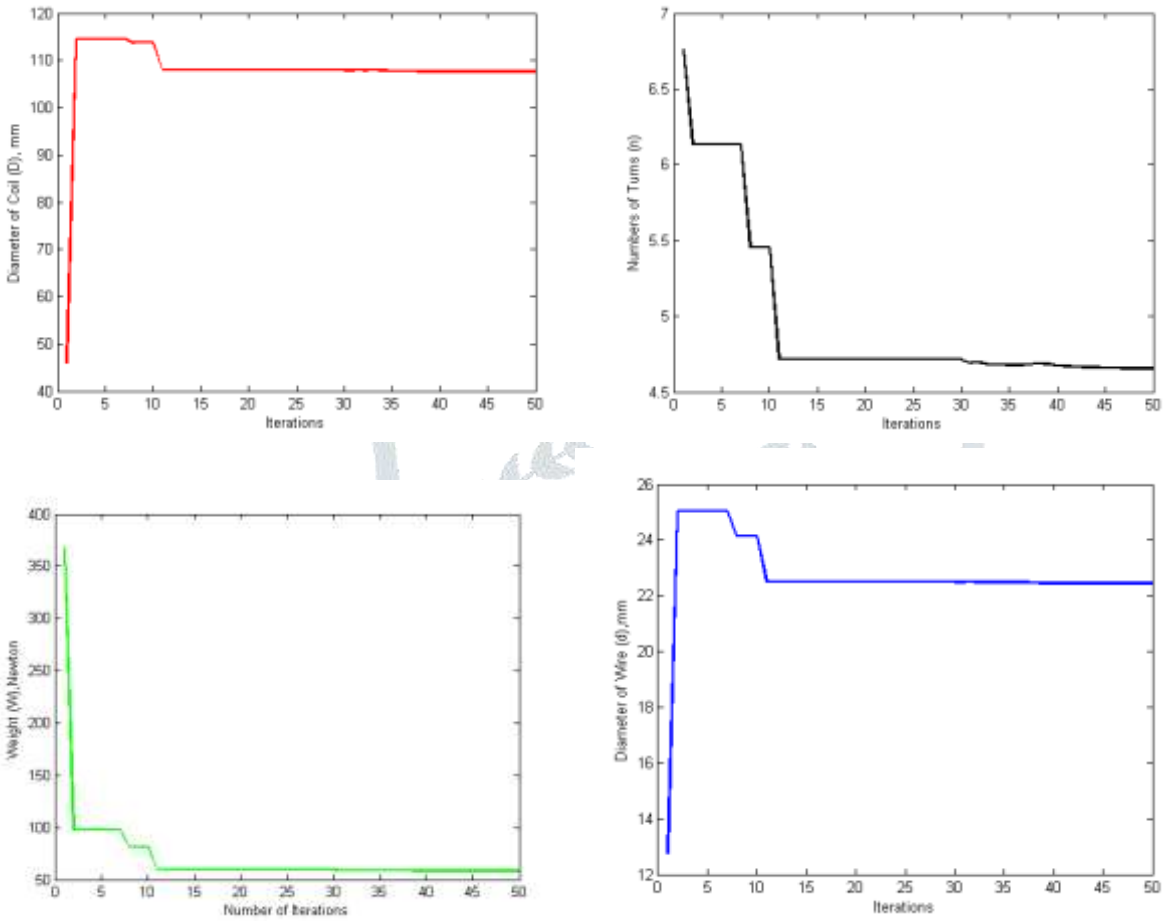

Fig 4 Optimized Results by Particle Swarm Optimization

Table 4 Comparison of optimized results by genetic algorithm and PSO

Method selected	**d**	**D**	**n**	**Weight**
With GA	22.5122	108.1916	5	59.0119
With PSO	22.4444	107.6658	5	58.3510

III. DISCUSSION AND CONCLUSION

A little consideration in table 4 shows that PSO give better results than GA but there is not much difference but when we observe weight minimization with iteration fig 3 and fig 4 then there in GA results optimized after 30 iterations but in PSO it is optimized in 15 iterations. There are less constraints in helical spring problem so there is less difference in results by GA and PSO but if constraints increase then than results evaluated by PSO will be better than GA and take less time also by PSO.

REFERENCES

[1] G.K. Agrawal, "helical torsion spring for minimum weight by geometric programming" journal of Optimization theory, 1978, 307-311

[2] Dr.ShapourAzarm, 1996, "Helical compression spring Optimization via Consol-optcad", University of Maryland, USA.

[3] David E.Goldberg., 1989, "Genetic Algorithms in search optimization a Machine learning", Addison- Wesley publishing company, Singapore

[4] Ponsich A., C. Azzaro-Pantel , S. Domenech, L. Pibouleau, "Constraint handling strategies in Genetic Algorithms application to optimal batch plant design" Laboratoire de G´enie Chimique de Toulouse, 5 rue Paulin Talabot BP 1301, 31106 Toulouse Cedex 1, France

[5] Razvan Cazacu, Lucian Grama, Mihai Dan Groza, "Shape and Size Optimization of A Gearbox Bracket Using Genetic Algorithms", Annals of the Oradea University, Fascicle of Management and Technological Engineering, Issue 1, May 2015

[6] David E. Goldberg "Genetic Algorithms", Pearson Education India, 2006.

[7] Richard Budynas, Keith Nisbett "Mechanical Engineering Design", Tata Mcgraw Hill Education Private Limited Eighth edition.

[8] Xueming Yang, Jinsha Yuan, Jiangye Yuan, Huina Mao, "A modified particle swarm optimizer with dynamic adaptation" Applied Mathematics and Computation 189 (2007) 1205–1213.

[9] K. Venayagamoorthy, "improving the performance of particle swarm optimization using Adaptive critics designs" Real-Time Power and Intelligent Systems Laboratory, USA