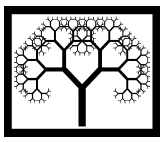
Paper 68



Proceedings of the Eleventh International Conference
on Computational Structures Technology,
B.H.V. Topping, (Editor),
Civil-Comp Press, Stirlingshire, Scotland

A Comparison of Genetic Algorithm and Particle Swarm Optimisation for Theoretical and Structural Applications

Z. Wang, T.J. McCarthy and M.N. Sheikh
Faculty of Engineering
University of Wollongong, Australia

Abstract

Genetic algorithms (GA) and particle swarm optimisation (PSO) are well-known for their ability in obtaining global optima. Some evidence exists in the structural engineering literature that PSO involves less overall computation effort than GA. Hence, these two methods have been selected and benchmarked against each other to test their relative robustness and efficiency for structural optimisation applications. This paper examines the performance and efficiency of these two optimisation algorithms in solving both mathematical benchmark functions and the classical ten-bar truss redundant problem. Tests are performed to assess the performance of each in relation to population size required and number of generations to achieve convergence. For the more complex problems, the PSO is shown to outperform the GA for smaller population sizes.

Keywords: optimisation, evolution, genetic algorithm, particle swarm optimisation, efficiency, performance comparison, population size, truss.

1 Introduction

In structural engineering optimisation, searching for the global optimum essentially means finding the best solution in a set of admissible solution candidates subject to one or more constraints. Optimisation techniques can be categorised into two groups. The first is classical optimisation which relies on derivative functions and has applications in many engineering and mathematical problems, especially where the region of the optimum can be predicted. Where the solution space is complicated or non-differentiable, such methods tend to converge on local optima rather than a global best solution. In these cases, the second group, non-derivative and evolutionary methods have been shown to succeed.

In this paper, two of the non-derivative evolutionary global optimisation search approaches are compared for both standard test mathematical functions and a much cited engineering design test application. To compare GA with PSO for obtaining global optima, two mathematical benchmark functions are taken from Yao et al. [1] and Premalatha and Natarajan [2]. The classical 10 bar redundant truss problem is also used [3]. In the mathematical test function benchmarking, GA and PSO are used to solve non-constraint problems within a limited range. Optimised solutions are compared with theoretical solution of those mathematical functions. For the engineering design test case, the classical redundant ten-bar truss problem is used with constraints modelled using penalty functions.

2 Genetic Algorithms (GAs)

The GA is an optimisation method based on the Darwinian rule of 'Survival of the fittest'. It simulates of the process of natural selection. A full coverage of GA can be found in Refs [4-5].

The GA deals with a population set of potential solutions. Each member of the population is coded to resemble a string of chromosomes. The representation is called the genotype. The unit make of the genotype is the gene. It represents the values of each design variable in a problem. There are various notations of chromosome representations such as binary bit string, floating points and q-ary coding [6].

Three main stages comprise the process of GA: fitness function evaluation; selection of the fittest individuals; and reproduction of the offspring [7]. In the first phase, fitness evaluation, the objective function is calculated for each of the population in a generation. The selection method employed in this paper is the tournament method [7]. In this, two members of the population are chosen at random and the better fitness is selected for reproduction. The probability of being chosen for the tournament is a function of the member's fitness. The reproduction phase has two operations. The main operation is called crossover, wherein the new generation is formed by combining part of the genes from two fit parents. Once the new population is created, a second operation called mutation is performed. This is only done on a small proportion of the population and is an attempt to maintain genetic diversity. The two operators create the next generation according to the formulae in Equation 1.

$$\begin{aligned} x &= \gamma x_{parent1} + (1-\gamma) x_{parent2} \\ x &= \alpha\beta\gamma x_{parent1} + \alpha\beta(1-\gamma) x_{parent2} \end{aligned} \tag{1}$$

where x is the design variable's representation, γ is a random number determining how many of the genes needed from that parent, α is the mutation probability and β is a random number between 0 and length of the chromosome to decide which gene is to be mutated or perturbed.

Processes of running GA's are as follows:

1. Randomly initialise the first population
2. Calculate the value of fitness for each member of the population
3. All members of the population go to the selection operator with their corresponding fitness values where the best are selected for reproduction.
4. Crossover operation occurs to produce the next generation.
    a. A (small) selection of the new population are mutated
5. Repeat step (2) to (4) until the convergence criteria are met.

The main choices the GA user has in using the technique are population size, selection method and mutation rate. Generally, the larger the population the more likely it is to find a near optimal solution

3 Particle Swarm Optimisation (PSO)

The PSO was developed using the analogy of social behaviour in a flock of birds flock or a school of fish. Kennedy and Eberhart [8] first introduced this evolutionary based optimisation technique. Similarly to GA, the fitness of an objective function is evaluated for each individual particle. At each iteration, the particle's movement within the swarm is influenced by three factors: it has an existing momentum and seeks to carry on its current course; its own best position from the first iteration to the current one acts as a draw to pull it back to a better state; it is also guided towards the best position in the swarm. These three components are added vectorally to produce the movement to its next location in the solution space where the objective function is recalculated.

An advantage of this algorithm is that it uses 'prior knowledge' in the search process. Information about the local particle and swarm are shared to create the swarming behaviour. The algorithm terminates when the swarm approaches the best known position regardless of the behaviour of the swarm, or the whole swarm converges to a position in the solution space.

Mathematically, the position x of a particle i, at time t is updated as

$$x_t^i = x_{t-1}^i + v_t^i \Delta t \tag{2}$$

where v_t^i is the velocity vector at time t, and Δt is the time step between iterations.

The incremental step, Δt, is taken as a unit (one) here for convenience. Particles have set values for each of the input variables which have been defined as the dimensions of the solution space. The velocity component of a particle has been separated into vectors that have multi-dimensions. Each of the dimensions in the velocity vector array represents the changing of a variable. The velocity vector of each particle is calculated as

$$v_t^i = wv_{t-1}^i + \frac{c_1 r_1 (p_{t-1}^i - x_{t-1}^i)}{\Delta t} + \frac{c_2 r_2 (p_{t-1}^g - x_{t-1}^i)}{\Delta t} \tag{3}$$

where, both r_1 and r_2 are generated uniformly between 0 and 1 for the purpose of providing randomness; p_{t-1}^i corresponds the best particle position of particle i in its time history; p_{t-1}^g represents the best position of all the particles at time t-1; c_1 and c_2 are acceleration parameters; w is the inertia coefficient.

The acceleration coefficients c_1 and c_2 control the random search effect of the cognitive and social components of velocity. The exploratory nature of particles is determined by the relative values of c_1 and c_2. A large cognitive acceleration coefficient, c_1, makes the particle wander excessively. A large social acceleration coefficient tends to trap the optimisation in local minima. Perez and Behdinan [9] claim that the coefficients, w, c_1 and c_2 are stable as long as the conditions in Equation (4) are met.

$$\begin{aligned} 0 < c_1 + c_2 < 4 \\ \frac{c_1 + c_2}{2} - 1 < w < 1 \end{aligned} \tag{4}$$

Instead of using static inertia weight, Eberhart and Shi [10] suggested a dynamic improvement can be made to the inertia weight by using a constriction factor. Ratnaweera [11] proposed a method that gave a higher value of c_1 initially and reducing it each iteration, while for c_2 initially had a low value and increasing it each iteration. They are given by:

$$\begin{aligned} c_1(t) &= \frac{(c_{1,\min} - c_{1,\max})t}{n_t} + c_{1,\max} \\ c_2(t) &= \frac{(c_{2,\max} - c_{2,\min})t}{n_t} + c_{2,\min} \end{aligned} \tag{5}$$

where $c_{1,max} = c_{2,max} = 2.5$ and $c_{1,min} = c_{2,min} = 0.5$, t is the time step and n_t is the total number of time steps. This method initially allows particles to explore the search space widely and then converges to a good optimum towards the end of the process. The controlling stopping condition for this is the maximum iteration number.

According to Perez and Behdinan [9], the process of PSO can be summarised into following steps:

1. Initialise a set of particles randomly distributed through the solution space. Initial velocities can be offset to zero or to random values.
2. Evaluate the objective function for the swarm of particles and store their position information.
3. Update the position of each particle from the corresponding velocity .
4. Repeat step (2) and (3) until the stop criterion is reached.

As with the GA, the user has to choose the population size. In addition the PSO momentum and acceleration coefficients need to be selected.

4 Optimisation algorithms in benchmark functions

The benchmark mathematical functions used to compare GA and PSO are summarised in Table 1. The purpose of the benchmarking is to test the robustness and efficiency of the optimisation method. The optimisation technique should display the following:

- It is able to search for and find the best solution in a problem.
- It has good computational efficiency.
- It requires minimal input from users and is less sensitive to these inputs.

Efficiency of optimisation methods is also important. The ability to find the global optimum of GA and PSO has been demonstrated by many researchers. It is useful to know which technique finds a solution with the smallest population and the least iterations.

Name	Functions	Dimension	range of x_i
Rosenbrock's Valley	$f(x)=\sum_{i=1}^{n-1}\left[100\left(x_{i+1}-x_i^2\right)^2+\left(1-x_i\right)^2\right]$	10	±2.048
Schwefel's Function	$f(x)=\sum_{i=1}^{n}\left[-x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)\right]$	10	±500

Table 1: Benchmark Functions[1-2]

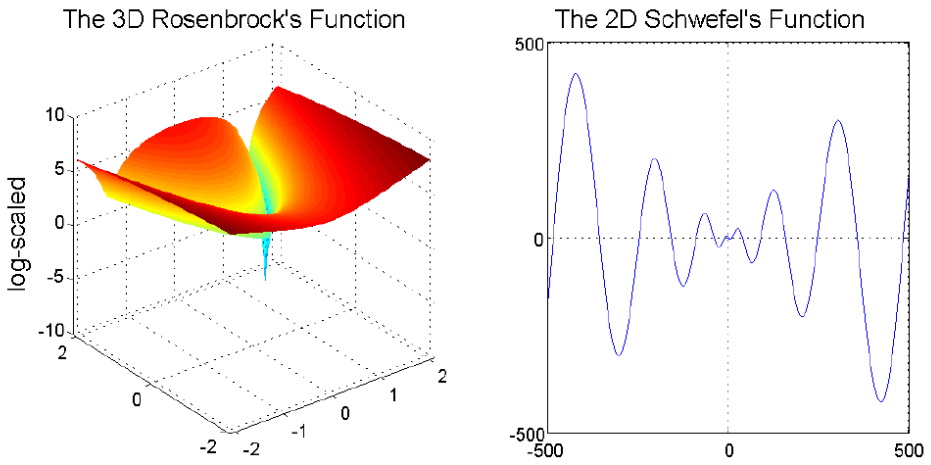


Figure 1: Graphs of simplified benchmark functions

4.1 Benchmark functions

The benchmark functions used have 10 dimensions. Rosenbrock's Valley function and Schwefel's function are uni-modal where there is only one minimum. The number of independent variables (dimensionality) defines the complexity of a

problem. The greater the number of design variables a problem has, the harder it is for an algorithm to find its optimum.

For illustration, Figure 1 shows the graph of the two benchmark functions in their smallest dimensional presentation. The 2D Schwefel function is presented in the *x*-*y* plot. The 3D Rosenbrock valley function has been plotted with a log-scale so that the global minimum can be seen. These functions are considered as good benchmark functions for an optimising program as they include several local minima and only one global minimum. Schwefel's function has the second best minimum geometrically distant from the global minimum. Some algorithms tend to get trapped in local optima.

4.2 Parameters

For the purpose of this study, the configurations of PSO and GA were fixed for all functions. Coefficients are given in Table 2. The Matlab global optimisation toolbox was employed as the GA engine for this study.

	PSO Configuration	**GA Configuration**
Population	20	20
Number of Iterations	100	100
Cognitive coefficient (c_1)	Equation (5)	N/A
Social coefficient (c_2)	Equation (5)	N/A
Inertia coefficient (w)	0.8	N/A
Selection method	N/A	Tournament
Crossover rate	N/A	0.8
Mutation	N/A	Adaptive feasible
Elite number	N/A	2

Table 2: PSO and GA configurable coefficients

It has been suggested by Wolpert and Macready [12] that within certain assumptions, there is no one algorithm that is the best for all problems. Furthermore, there is no one set of parameters for a single algorithm that is best for all problems either. Since both approaches has been proved to be successful in finding the global minimum for these benchmark functions in Yao et al. [1] and Premalatha and Natarajan [2], the main question here is on the computational efficiency. The performances of both GA and PSO are evaluated at the end of 100 iterations and also at convergence or 800 iterations. Following Ratnaweera's method [11] the PSO acceleration coefficients are taken from Equation (5) for this study.

4.3 Benchmark results and comparison

Rosenbrock's Valley function has a known global minimum of zero when all the x_i are equal to one. The Schwefel's function has its smallest value of -4189.829 when all the x_i equal to 420.9687[1-2]. Columns 3 and 4 of Tables 3 and 4 show the solutions found by GA and PSO after 800 iterations. The GA is slightly better than

PSO for Rosenbrock's function while both produce the correct solution for Schwefel's function. Since the ability to find the best solution in benchmark functions is not the main aim of this study, the computational efficiency is of interest. Therefore, results reported at the 100^{th} iteration are examined in terms of their performance and efficiency.

Experiments were carried out in two groups with respect to number of initial candidates (population). The initial population of the first group was generated uniformly on a random basis and is named Uniformly Generated Initial Population (UGIP) group. Each run had a different random initial population. The other group kept the same initial population, named as Same Initial Population (SIP) group, for both GA and PSO. This population was generated randomly for each run and given to both algorithms.

Tables 3 and 4 summarise the results of benchmark functions in comparison of both PSO and GA. The results of GA and PSO are taken from the best of 10 runs. The true minimum of these functions are given as well for reference.

	Rosenbrock's Valley Function						
	True	UGIP (800 iterations)		UGIP (100 iterations)		SIP (100 iterations)	
		GA	PSO	GA	PSO	GA	PSO
f(x)	0	**0.000811**	**0.02084**	0.183782	1.562608	1.32283	6.62202
x_1	1	**0.97239**	**0.99941**	0.989858	0.992501	0.990379	0.751846
x_2	1	**0.913261**	**0.999219**	1.00736	0.988137	0.980376	0.546456
x_3	1	**0.849115**	**0.998002**	1.002535	0.98474	0.969381	0.310197
x_4	1	**1.006635**	**0.995507**	0.991605	0.965107	0.940484	0.105409
x_5	1	**1.00165**	**0.991558**	0.980749	0.896334	0.888687	0.02242
x_6	1	**0.863724**	**0.983242**	0.970574	0.776144	0.788604	0.005274
x_7	1	**1.022321**	**0.967305**	0.954449	0.640092	0.617933	-0.01101
x_8	1	**0.923904**	**0.936416**	0.922452	0.425128	0.383442	0.007974
x_9	1	**0.889392**	**0.877236**	0.844496	0.192532	0.150443	0.021963
x_{10}	1	**0.563994**	**0.768571**	0.71227	0.039914	0.026062	0.018382

Table 3: Results comparison for Rosenbrock's Valley Function

Figures 2 and 3 show the global minima versus iterations from both GA and PSO for the two benchmark functions. Combining the final results shown in Tables 3 and 4, and the global best convergence patterns in Figure 2 and Figure 3, it can be concluded that each algorithm is capable of finding the near optimal solutions. The initial path to convergence is less clear.

The GA results for Rosenbrock's Valley are better at the end of the 100th iteration for both UGIP and SIP groups. In contrast, PSO showed a better fitness value for Schwefel's function. There are still significant errors between the true solution and the answers after 100 iterations. Examining the curves in Figure 2, it is arguable that the performance may be influenced by the quality of the initial population.

Therefore, the same experiments were carried out with the same initial population of both GA and PSO and shown in Figure 3. The PSO starts well for Rosenbrock's function but the GA ultimately overtakes it. The opposite is true for Schwefel's function solution.

	Schwefel's Function						
	True	UGIP (800 iterations)		UGIP (100 iterations)		SIP (100 iterations)	
		GA	PSO	GA	PSO	GA	PSO
$f(x)$	-4189.83	**-4189.83**	**-4189.83**	-3306.75	-4026.08	-3045.19	-3566.97
x_1	420.9687	**420.9687**	**420.9687**	-489.513	426.1113	-298.849	-500
x_2	420.9687	**420.9687**	**420.9687**	-277.859	422.9728	-142.741	423.218
x_3	420.9687	**420.9687**	**420.9687**	459.187	415.4945	-299.917	424.465
x_4	420.9687	**420.9687**	**420.9687**	409.1342	421.463	420.4999	421.707
x_5	420.9687	**420.9687**	**420.9687**	424.5183	-313.455	429.7488	423.777
x_6	420.9687	**420.9687**	**420.9687**	-294.442	417.4509	438.5725	421.623
x_7	420.9687	**420.9687**	**420.9687**	409.2465	420.1107	-297.189	431.479
x_8	420.9687	**420.9687**	**420.9687**	423.1055	429.7586	423.9826	429.473
x_9	420.9687	**420.9687**	**420.9687**	420.2371	430.312	-24.5423	-500
x_{10}	420.9687	**420.9687**	**420.9687**	423.2134	422.147	416.802	-299.33

Table 4: Results comparison for Schwefel's Function

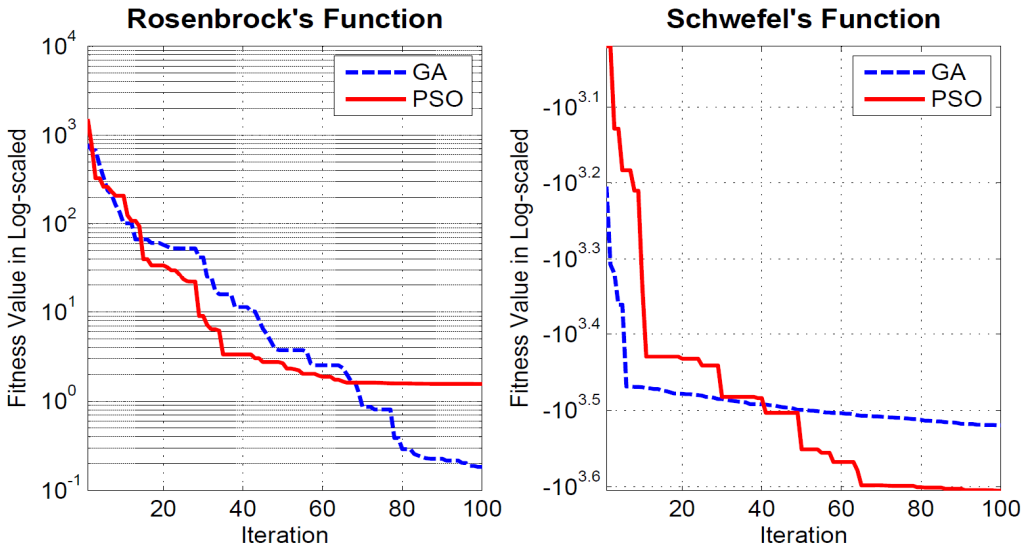


Figure 2: GA & PSO performance comparison with random initial candidates

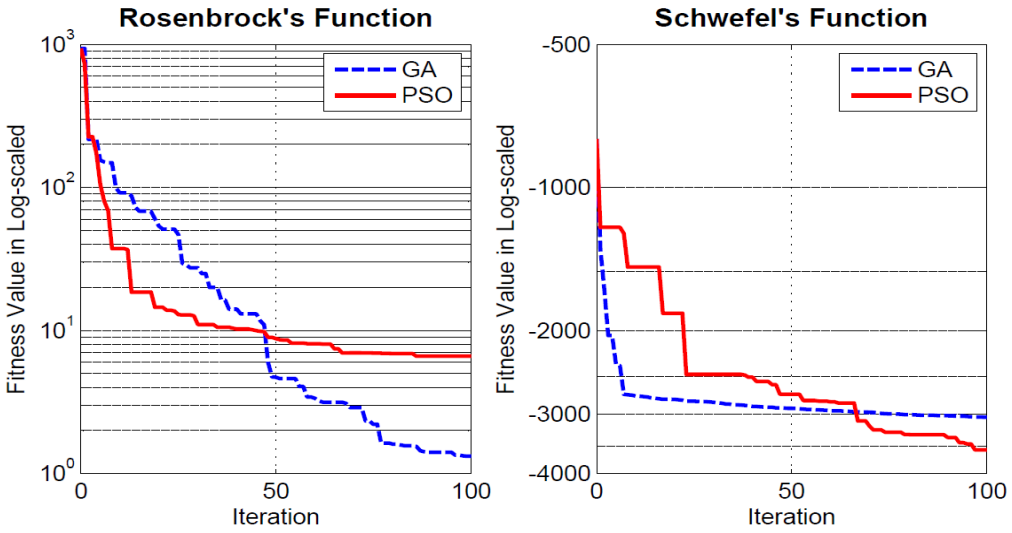


Figure 3: GA & PSO performance comparison with the same initial candidates

4.4 Benchmark results discussion

From the results above it cannot be concluded that one optimisation algorithm is better than the other. Other mathematical functions from Yao et al [1] and Premalatha and Natarajan and [2] were also investigated with similar results.

The curves in Figure 2 and 3 show only the best out of total ten runs. Taking the average performance and efficiency into consideration, Table 5 gives the mean and standard deviation values of the ten runs of each benchmark functions. The PSO demonstrates lower mean values and standard deviations. In other words, within the same amount of calculations, the lower mean value represents better efficiency. The lower standard deviation values indicates that the PSO has a more reliable performance for the early stage of the optimisation.

Fn Name	**UGIP** (100 iterations)			**SIP** (100 iterations)		
	Method	**Mean**	**Std. Dev**	**Method**	**Mean**	**Std. Dev**
RV	GA	32.52	33.80	GA	45.49	44.12
	PSO	**10.23**	**3.89**	PSO	**9.10**	**1.77**
SF	GA	-2623.8	421.0	GA	-2478.3	374.8
	PSO	**-3352.2**	**327.7**	PSO	**-3228.8**	**259.1**
Note: RV, Rosenbrock's Valley Function, SF, Schwefel's Function; UGIP, Uniformly Generated Initial Population group, SIP, the Same Initial Population group.						

Table 5: GA & PSO results comparison based on 10 runs

5 Ten-bar redundant truss problem

Since both GA's and PSO are evolutionary global solution search algorithms and it is difficult to judge their performance in purely mathematical functions, it is of interest to compare their effectiveness on a structural optimisation problem. The ten-bar redundant truss optimisation problem (Figure 4) was selected for this purpose. The structural optimisation problem selected herein is a classical problem in optimisation. Many researchers such as Galante [13], Jingui et al [14], Rajeev et al [15] and Leite et al [16] published their works about this problem. Ward and McCarthy applied a hybrid GA-ANN to solve this truss [17] while McCarthy and Fenwick [18] used a GA to optimise the topology of this truss.

5.1 Ten-bar truss redundant problem

The ten-bar redundant truss problem, shown in Figure 4, may be solved as a continuous problem or a discrete one. The continuous method assumes that the all the members of the truss have continuous range of selection of cross-section. The optimisation has been solved as a continuous problem by both Perez and Sunar [3, 9], where the cross section area varies between 0.1 $in.^2$ (64.52 mm^2) and 35.0 $in.^2$ (22580.6 mm^2).

The truss is to be optimised for minimum weight subject to constraints. In Figure 4, the solid circles are the truss nodes numbers, and the hollow circles are the member numbers. The stress for every member cannot exceed 25 ksi (172.3689 MPa), and the maximum deflection of any node is ±2 inch (50.8mm). The original problem was solved using imperial units. The current work was done in metric SI units (shown in Table 6).

Young's modulus (E)	68.9 GPa
Material density (ρ)	2770 kg/m3
Maximum allowable stress (σ)	172 MPa
Maximum allowable displacements (δ)	50.8 mm

Table 6: Material properties and constraints

This truss problem cannot easily be solved by conventional searching methods. The minimum weight goal conflicts with the two constraints of permissible stress and allowable deflection. This introduces the need for penalty functions to be applied to the objective function when constraints are violated. Final solutions must not violate either of the constraints.

It is not realistic to treat this truss problem as continuous since most steel truss members are made of predetermined sizes and provided in the form of manufacturer's handbook. For this reason, it is necessary to solve this optimisation as a discrete problem. Since most published works on the PSO [9, 19-20] algorithm use continuous structural applications, it is worthwhile to explore the performance

and effectiveness of PSO for discrete problems. Therefore, all member sections have been chosen from the American Institute of Steel Construction (AISC) Manual. All of the sections have been tabulated below (Table 7).

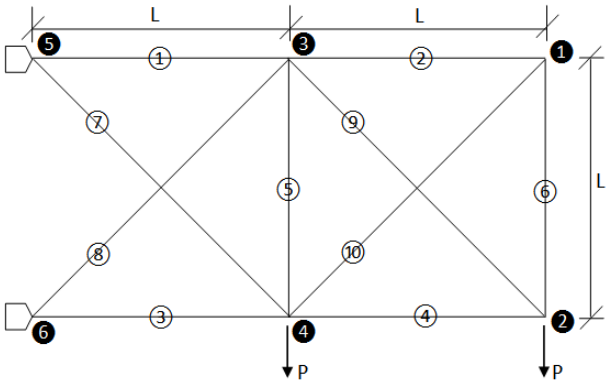


Figure 4: Ten-bar redundant truss

Section ID	Area (in.2)	Area (mm2)	Section ID	Area (in.2)	Area (mm2)	Section ID	Area (in.2)	Area (mm2)
1	1.62	1045	**15**	3.63	2342	**29**	11.50	7419
2	1.80	1161	**16**	3.84	2477	**30**	13.50	8710
3	1.99	1284	**17**	3.87	2497	**31**	13.90	8968
4	2.13	1374	**18**	3.88	2503	**32**	14.20	9161
5	2.38	1535	**19**	4.18	2697	**33**	15.50	10000
6	2.62	1690	**20**	4.22	2723	**34**	16.00	10323
7	2.63	1697	**21**	4.49	2897	**35**	16.90	10903
8	2.88	1858	**22**	4.59	2961	**36**	18.80	12129
9	2.93	1890	**23**	4.80	3097	**37**	19.90	12839
10	3.09	1994	**24**	4.97	3206	**38**	22.00	14194
11	3.13	2019	**25**	5.12	3303	**39**	22.90	14774
12	3.38	2181	**26**	5.74	3703	**40**	26.50	17097
13	3.47	2239	**27**	7.22	4658	**41**	30.00	19355
14	3.55	2290	**28**	7.97	5142	**42**	33.50	21613

Table 7: AISC truss member section table[19]

Converting from a continuous problem to a discrete problem for this particular truss example is simple. The conservative method of discretising is to round each member up to the next largest one from the AISC table and that is the approach adopted here.

5.2 Objective and penalty functions definition

The total mass of the truss includes a penalty mass which takes into account stress and/or displacement constraint violations. A number of trials were carried out to

determine suitable scale factors for the penalty values. The final values are given in Equations 6 and 7.

$$F_{obj} = \sum_{i=1}^{6} \rho A_i L + \sum_{i=1}^{4} \rho A_i \sqrt{2} L + P \tag{6}$$

$$P = \sum_{i=1}^{10} \Gamma_i \left(150^{(\sigma_1/\sigma)}\right) + \sum_{i=1}^{10} \Lambda_i \left(270^{(\delta_1/\delta)}\right) \tag{7}$$

where σ_i is the stress in a particular truss member, δ_i, is the deflection of a node in a particular direction, σ and δ are the maximum allowable stress and displacement respectively in Table 6. Γ=1 when $\sigma_i > \sigma$ and 0 when $\sigma_i \leq \sigma$, Λ = 1 when $\delta_i > \delta$ and 0 when $\delta_i \leq \delta$. Equation (7) is the penalty function used here to impose extra mass if any violations have been detected. The deflection penalty has a slightly heavier mass penalty. The values of 150 and 270 in Equation 7 were found by trial and error. The penalty threshold of Equation (7) has been set to assure there are no violations at the final solution.

5.3 Parameters and experiments configuration

The PSO and GA parameters used in the comparison are given in Table 8. For this problem, PSO cognitive acceleration coefficient c_1 and social acceleration coefficient c_2 are assigned as 2, the inertia coefficient w is 0.8. Static constants of these coefficients were used for the purpose of simplicity of PSO. The GA has a tournament selection algorithm with elite number of 2. This keeps the top two members of the population for the next generation without changing them.

To examine the efficiency of these two algorithms, population sizes of 10, 50, 100, 150 and 200 were used. The best results at iteration counts 100, 200, 400 and 800 were logged.

	PSO Configuration	**GA Configuration**
Population	10,50,100,150,200	10,50,100,150,200
Number of Iterations	100,200,400,800	100,200,400,800
Cognitive coefficient (c_1)	2	N/A
Social coefficient (c_2)	2	N/A
Inertia coefficient (w)	0.8	N/A
Selection method	N/A	Tournament
Crossover rate	N/A	0.8,
Mutation	N/A	Uniform
Elite number	N/A	2

Table 8: PSO and GA coefficients

5.4 Results from references

For this truss problem, there were a number of papers [13, 19-21] that have found the near optimum solution. The consensus is that the minimum weight is in the

region of 2490.6kg. Ward and McCarthy [17] solved the discrete problem with a combination of GA and ANN in 140 generations with a higher weight while McCarthy and Fenwick allowed for the removal of zero force members and achieve a weight of 2333kg [18]. A selection of results from the literature is given in Table 9.

The best solutions obtained from the current GA and PSO are included in Table 9. Both the GA and PSO are able to find the near optimum solution to the ten-bar redundant truss problem. Second, the PSO has slightly better configuration than that of GA in the particular sets of coefficients preset.

	Weight	**Truss member ID**									
	(kg)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
I.	2546.4	42	1	38	33	1	1	32	37	37	6
II.	2475.9	42	1	38	32	1	1	28	39	38	1
III.	2490.6	42	1	39	32	1	1	28	39	38	1
IV.	2490.6	42	1	39	32	1	1	28	39	38	1
V.	2702	42	1	39	33	2	1	29	40	39	1
VI.	2503.1	42	1	40	33	1	1	28	38	37	1
VII.	2498.7	42	2	39	33	1	2	28	38	38	1

Notes:

I. Rajeev and Krishnamoorthy [19]
II. Galante [13]
III. Cai [20]
IV. Schmidt and Thierauf [21]
V. Ward and McCarthy (GA&ANN) [17] with no constrain violation
VI. The best results from current GA with no constraint violations
VII. The best results from current PSO with no constraint violations

Table 9: Comparison of optimum solutions

5.5 Benchmark results and comparison

The tool used here to analyse the ten-bar truss problem was developed by the authors in Matlab. The best of ten runs was selected to be the solution shown in Tables 10 to 13. The average fitness values of the ten runs were summarised in these tables as well.

The experiments were carried using a standard desktop computer comprising an Intel Core i7 CPU at 2.93GHz and 4GB of RAM with Windows XP operating system. The tabulated results reveal that both PSO and GA are capable of finding feasible solutions without any violations. Further comparison of these two methods shows that the PSO generally has better results than GA in both best solution and average solution. Figure 5 plots the average weight results for both GA and PSO with increasing population size. The weight is normalised to the best solution from literature. The closer the curve is to unity, the closer it is to the published optimum

of 2490 kg. The solid line group represents the performance curve of PSO, while the dashed line group shows that of GA. It is clear that all of the solid lines are underneath the dashed lines for each population size.

Figure 6 shows the computational performance of GA and PSO in terms of run-time. The PSO has a general saving of time within the same population and number of iterations. Since the method used to calculate the stress and strain of the truss is the same, the time difference showed in the diagram can be considered as the discrepancy of time required due to the algorithms themselves.

	Pop	B.W.	A.W.	P	M.T (s)	Truss member ID									
						1	2	3	4	5	6	7	8	9	10
GA	10	2593.5	2929.9	0	5.93	40	2	39	33	3	6	32	39	39	2
PSO	10	2552.4	2775.6	0	3.37	41	5	38	32	2	4	28	39	40	2
GA	50	2544.7	2770	0	23.92	41	1	41	31	2	1	29	38	36	3
PSO	50	2553.1	2644.4	0	12.48	41	18	40	34	2	2	27	39	28	2
GA	100	2529.9	2615.8	0	48.01	42	1	39	29	1	1	29	38	39	1
PSO	100	2521.9	2569.1	0	25.77	41	1	40	31	1	2	29	37	38	5
GA	150	2511.9	2576.8	0	72.83	41	1	39	31	1	1	29	38	39	1
PSO	150	2512.7	2545.4	0	37.82	41	2	40	34	1	2	28	38	38	2
GA	200	2541.1	2583	0	96.54	41	1	40	30	1	1	30	38	37	1
PSO	200	2507.8	2531	0	46.00	42	2	38	33	2	2	28	39	38	1
Note: Pop= number in population; B.W., best weight; A.W., average weight; P, penalty; M.T., mean time (s)															

Table 10: GA and PSO results comparison for maximum 100 iteration runs

	Pop	B.W.	A.W.	P	M.T (s)	Truss member ID									
						1	2	3	4	5	6	7	8	9	10
GA	10	2578.5	2811.3	0	12	40	7	40	36	1	4	30	38	36	3
PSO	10	2526.9	2697.8	0	6.42	41	4	38	30	1	2	29	39	39	2
GA	50	2616.3	2720.9	0	47.11	40	1	40	34	1	8	35	37	36	21
PSO	50	2515.8	2545.4	0	25.05	41	2	40	31	2	1	28	39	39	1
GA	100	2522.1	2620.1	0	94.97	42	1	40	32	1	2	29	37	37	2
PSO	100	2503.1	2519.0	0	43.60	41	2	39	35	1	1	28	39	39	1
GA	150	2523.3	2593.8	0	141.1	41	1	38	33	1	1	29	38	39	1
PSO	150	2500.0	2515.8	0	74.48	42	2	39	33	1	1	28	38	38	2
GA	200	2518.6	2569.5	0	185.9	42	1	40	31	1	1	29	37	37	3
PSO	200	2502.2	2509.9	0	99.77	42	2	39	33	1	2	27	39	38	1
Note: Pop= number in population ; B.W., best weight; A.W., average weight; P, penalty; M.T., mean time															

Table 11: GA and PSO results comparison for maximum 200 iteration runs

	Pop	B.W.	A.W.	P	M.T (s)	Truss member ID									
						1	2	3	4	5	6	7	8	9	10
GA	10	2553.3	2920.8		23.28	41	1	40	30	1	2	31	38	37	1
PSO	10	2507.2	2647.8	0	12.81	41	2	39	35	1	1	28	39	39	2
GA	50	2577.0	2655.8	0	80.04	40	1	42	30	1	1	30	36	38	2
PSO	50	2512.9	2517.1	0	47.07	41	3	40	34	1	1	28	38	38	2
GA	100	2524.0	2592.5	0	155.1	42	4	39	31	1	6	28	39	38	4
PSO	100	2502.9	2513.0	0	94.54	42	2	39	33	1	2	28	38	38	2
GA	150	2527.2	2585.7	0	248.1	42	1	37	31	1	2	29	39	38	2
PSO	150	2499.5	2511.3	0	139.7	42	1	39	32	1	2	28	39	38	2
GA	200	2510.7	2562.7	0	331.2	41	1	38	34	1	1	29	38	38	1
PSO	200	2498.7	2505.3	0	185.9	42	2	39	33	1	2	28	38	38	1
Note: Pop = population size; B.W., best weight; A.W., average weight; P, penalty; M.T., mean time (s)															

Table 12: GA and PSO results comparison for maximum 400 iteration runs

	Pop	B.W.	A.W.	P	M.T (s)	Truss member ID									
						1	2	3	4	5	6	7	8	9	10
GA	10	2587.9	2766.3	0	46.08	41	4	38	29	1	4	33	39	39	1
PSO	10	2507.2	2572.1	0	25.66	41	2	39	35	1	1	28	39	39	2
GA	50	2529.9	2715.6	0	121.3	41	1	39	29	1	1	29	38	39	1
PSO	50	2502.4	2509.4	0	99.42	42	2	39	32	1	2	28	39	38	2
GA	100	2527.8	2597.3	0	435.0	41	1	39	30	1	1	28	39	40	1
PSO	100	2500.0	2505.4	0	193.0	42	1	39	32	1	1	27	39	39	2
GA	150	2503.1	2561.9	0	452.4	42	1	40	33	1	1	28	38	37	1
PSO	150	2501.9	2505.9	0	293.6	42	1	38	33	2	1	28	39	38	1
GA	200	2535.4	2578.3	0	563.3	42	1	38	31	1	1	30	38	37	2
PSO	200	2501.9	2507.3	0	384.8	42	1	38	33	1	2	28	39	38	1
Note: Pop = population size; B.W., best weight; A.W., average weight; P, penalty; M.T., mean time (s)															

Table 13: GA and PSO results comparison for maximum 800 iteration runs

5.6 Comparison discussion

In Tables 10-13 given above, the PSO has better average performance (average weight) in optimising the truss problem. As seen from Figure 5, the PSO group (solid lines) are located underneath of the GA. In the graph represented here, the lower the weight, the better of the performance. It shows that the PSO is better than GA.

- Both methods are sensitive to population size. The larger the number of candidates in the population the better the solution for both GA and PSO. PSO seems to perform better than GA for smaller populations. (See Figure 5).
- For a population of 100, the PSO seems very stable and converges in about 100 iterations.
- Running PSO requires less time than that of GA for the same problem. This can be shown in Figure 6, where the GA curve is always above the corresponding PSO curve. PSO uses half the time of the GA. There is a

potential reason for the GA taking longer than the PSO. The elitism operator used in the GA causes the algorithm to wait until all the population in a generation has been analysed to select the best pair.

- Within the same amount of computational load (i.e. the same population), the PSO can give better results than that of GA.

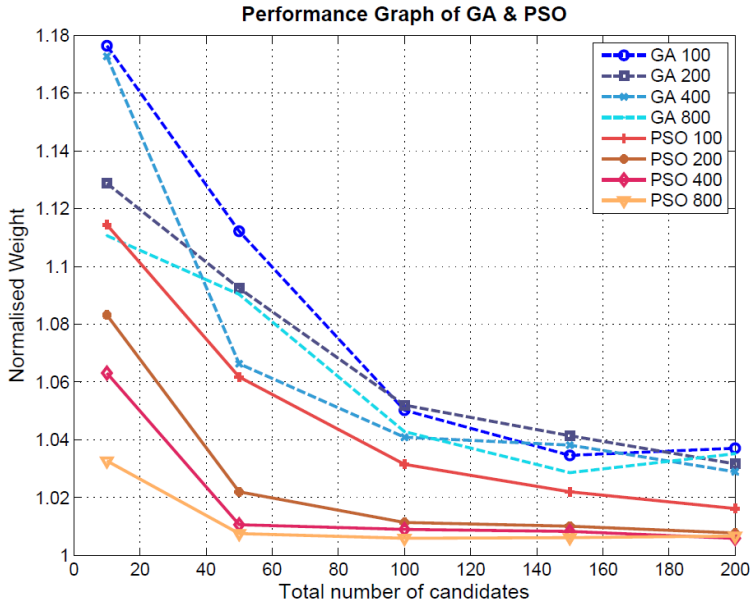


Figure 5 Optimisation results comparison for GA and PSO

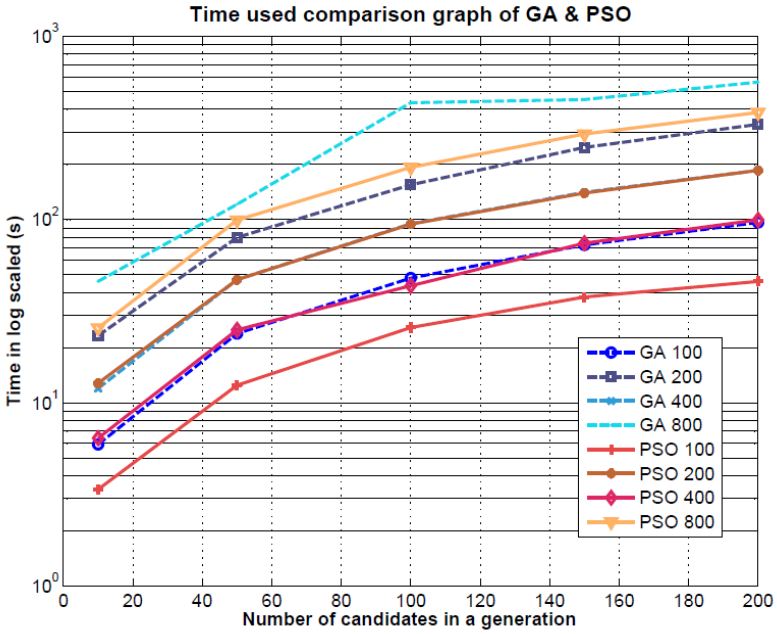


Figure 6 Time used comparison for GA and PSO

In terms of the final solution quality, the PSO has demonstrated a superior result for the same population and number of iterations. By being able to obtain good solutions with smaller populations and lower number of iterations, PSO will require fewer calls to the objective function. When such a function is computationally intensive, the number of calls will be critical.

Another conclusion that can be drawn from Figure 5 is the reliability of these two approaches. All the solid lines (PSO) show an expected behaviour with the increasing number of candidates and termination number. In contrast, the dashed lines (GA results curve) were less predictable. For example, the GA with the 50 population has a worse solution after 800 iterations than for 400 iterations.

6 Conclusion

This paper demonstrates that both GA and PSO have good ability of finding feasible and near optimum solutions without much difficulty. For the uni-modal mathematical benchmark functions, there is little difference between the GA and PSO performance. However, for the constraint based structural problem, there is a significant difference in the average performance between GA and PSO. The PSO appears to find good solutions with less computational effort than the GA.

The accuracy of the solutions from GA improves with increasing population size. The PSO seems to function well with smaller population size but benefits from increased number of iterations. Finally, the PSO algorithm appears to be more efficient than the GA.

References

[1] X. Yao, et al., "Evolutionary programming made fast," IEEE Transactions on Evolutionary Computation, vol. 3, pp. 82-102, 1999.

[2] K. Premalatha and A. M. Natarajan, "Combined heuristic optimization techniques for global minimization," Int. J. Advance. Soft Comput. Appl., vol. 2, pp. 85-99, 2010.

[3] M. Sunar and A. Belegunda, "Trust region methods for structural optimization using exact second order sensitiviey," International journal for numerical methods in engineering, vol. 32, pp. 275-93, 1991.

[4] D. Goldberg, Genetic algorithms in search, optimization, and machine learning. New York: Addison-Wesley, 1989.

[5] M. Gen and R. Cheng, “Genetic algorithms and engineering design”. New York: Wiley-Interscience, 1997.

[6] C. R. Reeves and J. E. Rowe, “Genetic algorithms - principles and perspectives”. London: Kluwer academic publishers, 2003.

[7] R. L. Tanaka and C. A. Martins, "Parallel dynamic optimisation of steel risers," Journal of offshore mechanics and arctic engineering, vol. 133, pp. 11302-9, 2011.

[8] J. Kennedy and R. Eberhart, "Particel swarm optimization," IEEE international conference on neural networks, vol. IV, pp. 1942-8, 1995.
[9] R. E. Perez and K. Behdinan, "Particle swarm approach for structural design optimization," Computers and Structures, vol. 85, pp. 1579-88, 2007.
[10] R. Eberhart and Y. Shi, "Comparing inertia weights and construction factors in particle swarm optimization," presented at the IEEE congress on evolutionary computation (CEC 2000), San Diego, CA, 2000.
[11] A. C. Ratnaweera, et al., "Particle Swarm Optimiser with Time Varying Acceleration Coefficents," presented at the International Conference on Soft Computing in Intelligent Systems, 2002.
[12] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimisation:," Evol. Comput., vol. 1, pp. 67-82, 1997.
[13] M. Galante, "Genetic algorithm as an approach to optimize real-word trusses," Int J Numer Meth Eng, vol. 39, pp. 361-82, 1996.
[14] L. Jingui, et al., "An improved strategy for GAs in structural optimisation," Computers and Structures, vol. 61, pp. 1185-191, 1996.
[15] S. Rajeev and C. S. Krishnamoorthy, "Genetic algorithms-based methodologies for design optimization of trusses," Journal of structural engineering, vol. 123, pp. 350-8, 1997.
[16] L. P. B. Leite and B. H. V. Topping, "Improved genetic operators for structural engineering optimisation," Advances in engineering software, vol. 29, pp. 529-62, 1998.
[17] K. Ward and T.J.McCarthy, "Fitness Evaluation for Structural Optimisation Genetic Algorithms Using Neural Networks," in Proceedings of the Fifth International Conference on Engineering Computational Technology, Gran Canaria, Spain, 2006, p. Paper 51.
[18] T. J. McCarthy and A. Fenwick, "Truss topology optimisation using genetic algorithms," presented at the 9th Int Conf Applications of Artificial Intelligence in Civil, Structural and Environmental Engineering, Malta, 2007.
[19] S. Rajeev and C. S. Krishnamoorthy, "Discrete optimiation of structures using genetic algorithm," Journal of structural engineering, vol. 118, pp. 1233-50, 1992.
[20] J. Cai, "Diskrete optimierung dynamisch belasteter tragwerke mit sequentiellen und parallelen evolutionssatrategien," Dissertation an der Universitt-Gesamthochschule-Essen, Essen, 1995.
[21] H. Schmidt and G. Thierauf, "A combined heuristic optimization technique," Advances in engineering software, vol. 36, pp. 11-19, 2003.