1-1-2009

A particle swarm optimisation approach to reinforced concrete beam design according to AS3600

Timothy J. McCarthy
University of Wollongong, timmc@uow.edu.au

Sean McCluskey
University of Wollongong, SMcCluskey@hatch.com.au

Follow this and additional works at: <https://ro.uow.edu.au/engpapers>

Part of the [Engineering Commons](#)

<https://ro.uow.edu.au/engpapers/529>

Recommended Citation

McCarthy, Timothy J. and McCluskey, Sean: A particle swarm optimisation approach to reinforced concrete beam design according to AS3600 2009, 1-14.
<https://ro.uow.edu.au/engpapers/529>

A Particle Swarm Optimisation Approach to Reinforced Concrete Beam Design according to AS3600

T.J. McCarthy and S. McCluskey
Faculty of Engineering
University of Wollongong, Australia

Abstract

A program has been developed to utilise the Particle Swarm Optimisation Algorithm for the cost optimum design of reinforced concrete beams. Multiple constraints according to Australian Standard 3600 have been implemented along with costing information. A series of tests were conducted to investigate the effect of different parameters of the particle swarm algorithm and some refinement of these parameters was conducted. The value penalty coefficients had a significant effect on the results, causing non-convergence when very large penalty coefficients were used, and convergence on non-viable results when very small penalty coefficients were used.

Keywords: reinforced concrete beam, particle swarm optimisation, PSO, penalty functions, visual basic for applications, VBA.

1 Introduction

The optimisation of reinforced concrete beams (RCBs) is a multi-variable problem due to the different effects, such as bending and shear, as well as the reinforcement conditions that are required to resist these forces. The process becomes even more involved when the costs of each material are considered, as the ratio of steel volumes to concrete volumes affect the final solution. Optimisation of RCBs on small scale projects may not be financially viable due to the time taken for optimisation compared to the potential savings, however on large scale projects where the same design may be used several times, the savings compound and the optimisation is viable.

A range of optimisation techniques have been applied in the past to structural engineering problems. Evolutionary algorithms such as the genetic algorithm (GA) have been applied to the search for optimal solutions to multivariate problems with both hard and soft constraints. A common structural optimisation problem that has been studied in the past is the 10 bar indeterminate truss. Rajeev and

Krishnamoorthy [1] demonstrated the utility of GA for the truss problem. This was extended by McCarthy and Fenwick [2] who used the GA to not only optimise the 10 bar truss but search for the optimal topology.

A number of authors have developed similar optimisation strategies using either GA or other evolutionary programming methods. One such method is the Particle Swarm Optimisation (PSO) algorithm. PSO is a search algorithm that is based on a population of random solutions, referred to as particles. Each particle represents a potential solution. The particles have a velocity, which moves them throughout the search space. These velocities are dynamically adjusted according to historical behaviours of the particles. This adjustment of velocity forces the particles to move in a common direction. The PSO algorithm was first designed to simulate birds seeking food [3]. PSO was first introduced in 1995, and since then, many changes have been made to improve convergence of the PSO and increasing the diversity of the swarm. [4]

The aim of this paper is to apply PSO to the design RCBs under user defined loads and to investigate the sensitivity of the solution to the various PSO

2 PSO Algorithm

2.1 Particle position and velocity

There are several parameters that are used to describe and control the behaviour of a swarm. The swarm has a population of particles. Each particle has a position in the solution space and a velocity vector which moves it to a new position. If the position of the i^{th} particle at step, t , is $x_i(t)$, then the position at step $t+1$ is determined from

$$x_i(t+1) = x_i(t) + v_i(t) \quad (1)$$

where $v_i(t)$ is the velocity at step t .

The velocities of the particles are determined by the best position history of particles. The two positions that are used are the personal best and the neighbourhood best. The personal best position is the best solution that each individual particle has achieved during the simulation. The neighbourhood best position is the best position that any particle in the swarm has achieved during the simulation. These best positions serve as the attractor for the particle and when combined with rules to adjust positions, the particles change position and converge on the optimum position.

The velocity is made up of three vector components. The first is a momentum or inertia component which is based on the particle's velocity at the previous time step. This gives a particle the tendency to continue on its current course. The second is a cognitive or memory component which is derived from the particle's own best position during all the iterations so far. This component attracts the particle towards its own previous best position in the solution space. The final component is the social or swarm component. The particle is attracted towards the best position found for the entire swarm. Each of these components are controlled by constant factors

and random modifiers. For the “*global best*” algorithm the velocity at step t is given by Eq. (2).

$$v_{ij}(t) = mv_{ij}(t-1) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] + c_2 r_{2j}(t)[\hat{y}_j(t) - x_{ij}(t)] \quad (2)$$

where v_{ij} is the velocity of the i^{th} particle in the j^{th} dimension, m is the momentum constant, c_1 and c_2 are acceleration constants while $r_{1j}(t)$ and $r_{2j}(t)$ are random numbers assigned at each time step. The position vector $y_{ij}(t)$ is the best position to date for the i^{th} particle and $\hat{y}_j(t)$ is the best position to date for the whole swarm.

Figure 1 Velocity components (after [4])

2.2 PSO control parameters

The correct selection of the PSO parameters has a marked effect on the performance of the algorithm. These parameters are swarm size, inertia coefficient and acceleration coefficient. For local best algorithm the neighbourhood size is also required. In this paper the global best algorithm is used.

2.2.1 Swarm size

If a high number of particles is selected, the PSO algorithm can explore a greater area in each iteration, and has a greater chance of finding the global optimum. If the swarm size is too large, it can result in parallel random search, as well as increases in computational time. Empirical studies have shown that the PSO algorithm can find the optimum with swarm sizes of 10 to 30 particles [5,6]. These values are heuristic and although they serve as a good guide, the best number of particles is dependant on the optimisation problem.

2.2.2 Acceleration coefficients

The acceleration constants c_1 and c_2 , when combined with the random numbers r_1 and r_2 , control the random search effect of the cognitive and social components of velocity. The relative values of c_1 and c_2 determine the exploratory nature of the particles. If $c_1 \gg c_2$, the particle will tend to wander excessively, while if $c_2 \gg c_1$,

the particles will converge too quickly, becoming trapped in local minima. One method of determining the values of c_1 and c_2 , proposed by Ratnaweera [7], involves having a higher value of c_1 initially and reducing it each iteration, and having a low initial value of c_2 and increasing it each iteration. The values of c_1 and c_2 are given by Eq. 3.

$$\begin{aligned} c_1(t) &= (c_{1,\min} - c_{1,\max}) \frac{t}{n_t} + c_{1,\max} \\ c_2(t) &= (c_{2,\max} - c_{2,\min}) \frac{t}{n_t} + c_{2,\min} \end{aligned} \quad (3)$$

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 process allows the particles to explore the search space initially, and then convergence to good optimum towards the end of the optimisation process. This method assumes that the number of iterations is a terminating value.

2.2.3 Initialisation of the PSO

Initialisation of particles is the first step involved in the PSO. The parameters discussed above need to be defined, as well as the initial and personal best positions and initial velocities of the particles. Particle positions are initialised to cover the search space uniformly, as this uniformity is a heavy influence on the efficiency of the PSO algorithm. If an optimum result occurs in a region that is not covered by the initial particle spread, the only way for particles to find this optimum is if a particle wanders into it due to its momentum.

A way to set the positions of particles within the domain limits $x_{j,\min}$ and $x_{j,\max}$ is:

$$x_{ij} = x_{j,\min} + r_{ij} (x_{j,\max} - x_{j,\min}) \quad (4)$$

where r_{ij} is a random number between 0 and 1 and j is the dimension being set.

The initial velocity can be set to zero, $v_i = 0$. This will cause initial movement of particles to be influenced only by the cognitive and social components of velocity. Initial velocities can be randomly set, although it is not required, as the random position initialisation ensures random movement. Also, if random velocities are set and are too large, the momentum and movements of these particles may force them out of the search space, and cause the algorithm to take a greater number of iterations to converge.

2.2.4 Stopping conditions

If the PSO is terminated too early, a suboptimal solution will be obtained, while if the PSO is terminated too late, the extra iterations take up computational time. There are a number of stopping conditions:

- when a maximum number of iterations has been exceeded.
- when no improvement occurs over a number of iterations.
- when a certain percentage of the swarm clusters around an optimum.
- when the objective function slope is approximately zero.

3 Optimisation of Reinforced concrete beams

3.1 Background

There are several different ways that the design of RCBs can be optimised. One aspect that can be optimised is the mass of the beam by finding the optimum ratio between steel and concrete to reduce the overall mass of the RCB. A common objective for optimising RCBs is reduction of cost. Studies have been conducted by Al-Salloum and Siddiqi [8] and Leps and Sejnoha [9] into cost optimisation of RCBs utilising different optimisation techniques. These studies have been based on different design standards. Al-Salloum's method involved non-iterative Pareto graphs with performance constraints based on the ACI code. The objective function used was based on the cost of concrete, steel reinforcement and formwork. Leps and Sejnoha [9] developed a genetic algorithm based optimisation that is modified to include a simulated annealing function. This was done because in their previous studies, Leps and Sejnoha found that simple genetic algorithms were suitable when the variables were limited to the concrete dimensions and longitudinal reinforcement area, however did not converge when the shear reinforcement was included.

3.2 PSO for reinforced concrete beams

The optimisation process involves developing an objective function that is subject to a number of restraints. For the RCB, the objective function will be the total cost of the beam, and the constraints will be limitations of bending strength, as well as practical size constraints. The development of the objective function has been done using Microsoft Excel Macros and Visual Basic for Applications (VBA).

3.2.1 Objective function

The program developed for this research optimises the RCB design for cost. The objective function to be optimised is

$$C_T = (bDC_C + A_{st}C_{st} + (b + 2D)C_F) \times L + \sum p_i k \quad (5)$$

where the costs are: C_T = Total cost of RCB (A\$/m), C_C = Cost of concrete (A\$/m³), C_{st} = Cost of tensile reinforcement (A\$/t), C_F = Cost of formwork (A\$/m²). The structural parameters are: b = beam width (m), D = total beam depth (m), A_{st} = Area of tensile reinforcement (m²) and L = Beam Length (m).

The constraints from AS3600 [10, 11] are applied as a penalty function with p_i = penalty coefficient and k = penalised value. The design checks here are based only on moment capacity.

For this problem, the design variables are the beam width (b), the beam depth (D), tensile reinforcement area (A_{st}). All other values in the objective function are constants that are defined by the user. It should be noted that the beam depth D is the depth to the reinforcement (d) plus the concrete cover, which is defined by the user. The penalty coefficients and penalised values are defined in the next section.

3.2.2 Penalty function

A set of penalty functions where applied to the objective function based on the applied constraints. These penalty functions increased the value of the penalty function by multiplying the difference between the allowable and actual value by a set coefficient. These coefficients where determined by experimentation.

The following penalty functions where applied:

Moment capacity

$$M_{pen} = (M^* - \phi M_{uo}) \times p_M \dots \text{if} \dots \phi M_{uo} < M^* \quad (6)$$

where M^* is maximum applied moment, M_{uo} is the AS3600 ultimate flexural strength and $\phi=0.8$, a partial safety factor.

Impractical proportions

$$R_{pen1} = \begin{cases} 0.25D - b \dots \text{if} \dots b < 0.25D \\ b - 10D \dots \text{if} \dots b > 10D \end{cases} \times p_R \quad (7)$$

Excessive depth

$$D_{pen} = (D - D_{max}) \times p_D \dots \text{if} \dots D > D_{max} \quad (8)$$

Excessive width

$$b_{pen} = (b - b_{max}) \times p_b \dots \text{if} \dots b > b_{max} \quad (9)$$

Non ductile design

$$ku_{pen} = (ku - 0.36) \times p_{ku} \dots \text{if} \dots ku > 0.3 \quad (10)$$

where k_u , the ductility factor, is the ratio of depth to neutral axis over effective depth of the beam.

The penalty for non-ductile design was implemented when $k_u > 0.3$, however values of k_u up to 0.36 where considered acceptable. The penalty coefficients for width and depth are increased linearly throughout the algorithms process to allow for initial coverage of the search space while restricting invalid results at the end of the algorithms process. The final objective function used for optimisation is given in Eq. 11.

$$C_T = (bDC_C + A_{st}C_{st} + (b + 2D)C_F) \times L + M_{pen} + R_{pen} + D_{pen} + b_{pen} + ku_{pen} \quad (11)$$

3.2.3 Swarm Initialisation

Each variable in the objective function was initialised by multiplying the allowable range of each variable by a random number between 0 and 1. This process allowed for coverage of the entire search space. The allowable range of each variable was determined from practical limitations and design strength requirements.

The allowable range of the beam width was set to a minimum of 100mm as a practical limitation, and the maximum allowable width as determined by the user. The initial position of width for each particle was determined by:

$$b_i = 100mm + (b_{\max} - 100mm) \times r \quad (12)$$

where b_i = particle beam width (mm), $b_{\max}$ = maximum beam width (mm) and r is random number between 0 and 1.

Similarly, the variables for depth and area of tensile steel were initialised. Minimum and maximum steel limits were chosen according to AS3600. Initial velocity vectors were all set to zero.

3.2.4 Position update

The PSO algorithm that was used was a global best PSO. This was chosen to allow a broad coverage of the search space, while converging on one solution.

The PSO algorithm assesses each of the initial particle positions based on the objective function (Equation 11). The initial position of each particle becomes its personal best, and the location of the particle with the best result becomes the global best position. The particle variables are then used in Equation 2, which gives the variable change for each particle based on the current location compared to the personal best and global best positions. These variable changes are then used in Equation 1 to determine the new particle position.

The new particle positions are then assessed based on the objective function (Equation 11). This new evaluation of the objective function is compared to the particles personal best and the global best positions. If the new assessment of the objective function is lower than the objective function value of the particles personal best, the new position becomes the particles personal best position. If the new assessment of the objective function is lower than the objective function value of the global best position, the new position becomes the global best position.

4 PSO Analysis

4.1 Parameters

A simple user interface was developed to input the constraints of the objective function and define the population size (Figure 2). A number of tests have been conducted to investigate the effects that different parameters of the PSO algorithm and the objective function have on the behaviour and output of the PSO program. The parameters tested were the social and cognitive acceleration coefficients, the momentum coefficient of the particle, the number of particles in the swarm, the ductility penalty coefficient and the moment capacity penalty coefficient. For each experiment, 10 test runs were conducted to analyse the variance of test results. Comparative costs for concrete, reinforcement and formwork were taken from Rawlinson's Australian Construction Handbook [13].

	A	B	C	D	E	F	G	H
1	Parameters							
2								
3	Component Costs				Material Strength			
4	Concrete Cost	306	\$/m ³		Concrete Strength	32	MPa	
5	Steel Cost	2400	\$/t		Steel Strength	500	MPa	
6	Formwork Cost	129	\$/m ²					
7								
8					Applied Moments			
9					M	122.5	kNm	
10								
11								
12								
13	Beam Limits				PSO Settings			
14	Minimum Cover	65	mm		# of particles		25	
15	Maximum Depth	1000	mm		# of iterations		2000	
16	Maximum Width	250	mm					
17	Beam Length	7000	mm					
18								
19								
20	Result							
21	Width (mm)	Depth (mm)	Ast (mm ²)	Cost (\$)	Run			
22	167.4508959	598.2124064	1347.386275	1234.862				
23								

Figure 2 User interface

Applied Moment, M*	150kNm
Concrete Cost	\$247.8/m ³
Reinforcement Cost	\$1765/t
Formwork Cost	\$125/m ²
p _D , p _b	\$1/mm
p _R	\$1000
p _M	\$100/Nmm
p _{ku}	\$5000
Momentum Coefficient	0.7
c _{1,min} , c _{2,min}	0.5
c _{1,max} , c _{2,max}	2.5

Table 1 PSO Initial parameters

4.2 Results for PSO parameters

4.2.1 Swarm size

The swarm size was varied to investigate the swarm size that gave the best performance for convergence and processing time. The average performance of the swarm for each test is summarised in Table 2.

The swarm size has minimal effect on the result produced, the number of iterations taken for convergence and the variation of iterations taken for convergence is reduced significantly between 5 and 25 particles. There is more unpredictability in the cost and the number of iterations for the smaller populations. The total number of objective function evaluations can be estimated from the average iterations and population size. From this a swarm size of 10 is very efficient. The swarm size of 25 gives a good balance between predictability and efficiency. This

has been used for the subsequent tests. This is within the range suggested previously [5,6].

Swarm size	Ave. Beam Cost (A\$)	St. Dev. of Beam Costs (A\$)	Min Iterations to Convergence	Max Iterations to Convergence	Average Iterations to Convergence	St. Dev. of Iterations
5	826.288	0.0148	167	750	312	172
10	826.280	1.02×10^{-5}	137	332	232	64
25	826.280	0	148	223	177	20
50	826.280	1.14×10^{-13}	129	176	154	16
100	826.280	0	119	144	132	9

Table 2 Results of Varying Swarm Size – Statistics for 10 runs at each population

4.2.2 Social Acceleration Coefficient

The social acceleration coefficient was varied to investigate the effect of the global best position on the swarm. By increasing the social acceleration coefficient, the swarm should be drawn towards an optimum sooner, decreasing the number of iterations to convergence, but reducing the searching capacity of the algorithm. The results from this test are summarised in Table 3.

$c_{2,min} / c_{2,max}$	Average Beam Cost (A\$)	St. Deviation of Beam Costs (A\$)	Min Iterations to Convergence	Max Iterations to Convergence	Average Iterations to Convergence	St. Dev. of Iterations to Convergence
0.25/1	826.280	8.47×10^{-14}	134	201	168	22
0.5/1.5	826.280	1.2×10^{-13}	145	196	163	17
0.5/2	826.280	1.14×10^{-13}	167	234	181	23
0.5/2.5	826.280	1.07×10^{-13}	120	217	156	33
1/3.5	826.280	1.96×10^{-10}	168	253	211	24

Table 3 Varying Social Acceleration Coefficient - 10 runs at each setting

The overall performance of the program is not significantly affected by varying of the social acceleration coefficient, except for when $c_{2,min} = 1$ and $c_{2,max} = 3.5$. Results from this test show a slight increase in the number of iterations to convergence, as well as an increase in standard deviation of the beam cost, although this is not significant. The reason for this increase in iterations is due to the tendency of particles to overshoot the global optimum when searching due to the increased velocity due to the social acceleration coefficient.

4.2.3 Variation of Cognitive Acceleration Component

By increasing the cognitive acceleration coefficient, the swarm should cover the search area more widely, increasing the number of iterations to convergence, while also increasing the searching capacity of the algorithm. The results from this test are summarised in Table 4.

$c_{1,min} / c_{1,max}$	Average Beam Cost (\$)	St. Deviation of Beam Costs (\$)	Min Iterations to Convergence	Max Iterations to Convergence	Average Iterations to Convergence	St. Dev of Iterations to Convergence
0.25/1	826.2878	0.0237	77	750	159	209
0.5/1.5	826.2803	1.65×10^{-13}	109	157	128	15
0.5/2	826.2803	1.07×10^{-13}	116	162	136	15
0.5/2.5	826.2803	1.0×10^{-13}	133	209	163	25
1/3.5	826.2803	9.86×10^{-9}	345	513	421	52

Table 4 Results of Varying Cognitive Acceleration Coefficient - 10 runs per setting

The results show that varying the cognitive acceleration coefficient had a minimal effect on the output on the result, however the performance of the program at the highest and lowest values of the cognitive acceleration coefficient was noticeably affected. When $c_{1,min} = 0.25$ and $c_{1,max} = 1$, the average beam cost was similar to all other test runs though the number of iterations to convergence was highly unpredictable. This result indicated that the search space was not adequately covered initially and the program converged on a local optimum. At the highest value of the cognitive acceleration coefficient, $c_{1,min} = 1$ and $c_{1,max} = 3.5$, convergence occurred after a much higher number of iterations. This was due to the particle behaviour causing each to search around its respective personal best position.

4.2.4 Variation of Momentum Coefficient

The momentum coefficient was varied to investigate the effect of searching capabilities on the particles based on previous changes. By increasing the momentum coefficient of each particle, the coverage of the search space is increased, although the possibility of divergence is increased. The results of this test are summarised in Table 5.

The results show that when the momentum coefficient is increased above 1, non-optimum results are produced. Also as the momentum coefficient was increased, the number of test runs with non-convergence also increased. The results indicated that the particles diverged dramatically after very few iterations. It was also noted that while some convergence occurred with these higher momentum coefficients, the particles still diverged after finding better results initially.

Momentum Coefficient	Average Beam Cost (\$)	St. Deviation of Beam Costs (\$)	Min Iterations to Convergence	Max Iterations to Convergence	Average Iterations to Convergence	St. Dev of Iterations to Convergence
0.5	826.2803	1.93×10^{-13}	92	163	121	23
0.7	826.2803	1.07×10^{-13}	129	209	175	26
1*	860.8871	46.899	7	56	29	15
1.25*	902.4413	65.444	5	19	8	5
1.5*	953.1978	83.286	8	16	10	3

* Some test runs did not converge

Table 5 Results of Varying Momentum Coefficient

There was little difference in results when the momentum coefficient is set to 0.5 or 0.7. Tests using a momentum coefficient of 0.5 converged at fewer iterations, however there was a slightly less variation of results for tests using a momentum coefficient of 0.7 and this value was used for the rest of the study.

4.3 Results for Penalty function parameters

4.3.1 General

Penalty functions are used to apply structural constraints such as adequate moment capacity, reasonable geometry and ductility limits. The scale of these functions can have a marked impact on the solution found by the PSO. In many cases, the PSO fails to find a viable solution if the penalty functions are inappropriately designed.

4.3.2 Variation of Moment Capacity Penalty Coefficient

The moment capacity penalty coefficient was varied to investigate how large the penalty coefficient was required to be to limit non-viable results, without being too restrictive on searching behaviour. The results from this test are summarised in Table 6.

Moment Capacity Penalty (\$/Nmm)		Width (mm)	Depth (mm)	A_{st} (mm ²)	ϕM_{uo} (kNm)	k_u	Cost (\$)	St. Dev. of Cost (A\$)
1	Best	121.33	485.31	898.80	150.0	0.341	826.28	2.01×10^{-11}
	Worst	121.33	485.31	898.80	150.0	0.341	826.28	
	Average						826.28	
0.01	Best	121.33	485.31	898.80	150.0	0.341	826.28	49.80
	Worst	140.60	565.65	1080.67	214.0	0.304	983.77	
	Average						842.03	
0.0001	Best	121.33	485.31	898.80	150.0	0.341	826.28	6.41×10^{-9}
	Worst	121.33	485.31	898.80	150.0	0.341	826.28	
	Average						826.28	
0.000001	Best	53.81*	215.22	194.75	14.2	0.376	321.45*	0.669
	Worst	54.13*	216.50	169.89	14.4	0.376	323.58*	
	Average						322.51*	

* No viable solution found

Table 6 Moment Capacity Penalty Coefficient

The results show that the moment capacity penalty coefficient does not need to be very high to maintain viable results. Consistent results were gained from all values of the moment capacity penalty coefficient above \$0.0001/Nmm. When this penalty function was omitted completely, no viable solutions were found.

4.3.3 Variation of Ductility Penalty Coefficient

The ductility penalty coefficient was varied to analyse the effect of restricting the allowable depth to neutral axis by limiting the allowable amount of reinforcement in the cross section. The penalty function came into effect when $k_u > 0.3$, however values of k_u up to 0.36 were viable as this is allowed by AS3600 [10]. Results from this test are summarised in Table 7.

Ductility Penalty (\$)		Width (mm)	Depth (mm)	A_{st} (mm ²)	ϕM_{uo} (kNm)	k_u	Cost (A\$)	St. Dev. of Cost (A\$)
10000	Best	121.33	485.31	898.80	150.0	0.341	826.28	147.56
	Worst	182.89	610.91	1476.1	316.9	0.295	1191.57	
	Average						916.442	
5000	Best	121.33	485.31	898.80	150.0	0.341	826.28	2.92×10^{-12}
	Worst	121.33	485.31	898.80	150.0	0.341	826.28	
	Average						826.28	
1000	Best	121.33	485.31	898.80	150	0.341	826.28	9.15×10^{-8}
	Worst	121.33	485.31	898.80	150	0.341	826.28	
	Average						826.28	
400	Best	114.54	458.16	990.21	150	0.422	788.41*	3.263
	Worst	116.47	465.89	898.80	150	0.396	798.87*	
	Average						793.82*	

* No viable solution found

Table 7 Varying Ductility Penalty Function

From the results, it can be seen that the value of the ductility penalty coefficient has a major influence over the results of the PSO program. When the value of p_{ku} was set at \$10 000, the program had trouble converging on a viable result. Between p_{ku} values of \$1 000 and \$10 000, the results remained constant, with the optimum result being obtained. As the value of p_{ku} was decreased to zero, non-viable results were obtained. A p_{ku} value between \$400 and \$1000 may produce viable results with a lower penalty function value, however it is believed that this may have adverse effects on the results under different loading and cost conditions.

4.3.4 Variable Dimension Ratio Penalty, R_{pen}

The dimension ratio penalty function was modified to see the effect of linearly increasing the dimension ratio penalty over the course of the PSO algorithm process. The tests were conducted with p_{ku} values of \$1 000 and \$10 000, based on results from previous testing. $p_{ku} = \$1\ 000$ was selected as this produced consistent results with a linearly increasing p_R value, and the effect of a constant p_R value was investigated. $p_{ku} = \$10\ 000$ was selected due to the program's difficulty converging on a result with a linearly increasing p_R value, so the effect of a constant p_R value was investigated. The results of this test are summarised in Table 8.

The results show that when the program had trouble converging at a higher p_{ku} value, results were improved with a constant dimension ratio penalty coefficient. When the dimension ratio penalty coefficient was varied linearly throughout the algorithm process, the program only converged on the optimum result for 50% of the test runs. This was increased to 80% when the dimension ratio penalty coefficient was constant. The variation of results was also decreased with a constant dimension ratio penalty coefficient.

The cause of the non converging results with a varied dimension ratio penalty coefficient is due to the small penalisation of non viable results at the beginning of the algorithm process. Results produced at the beginning of the process that would be penalised harshly with a constant penalty are deemed to be the global best position, so particles are drawn to this location, but the objective function value of

the position is increased with every iteration, so particles have converged on a position with an increasing objective function value. A constant dimension ratio penalty was chosen for other tests in the study.

P _R	P _{ku} (\$)		W mm	D mm	A _{st} (mm ²)	φM _{uo} kNm	k _u	Cost (A\$)	St. Devi.of Beam Cost (\$)
Varied	1000	Best	121	485	898.80	150.0	0.341	826.28	1.27x10 ⁻⁸
		Worst	121	485	898.80	150.0	0.341	826.28	
		Average						826.28	
Constant	1000	Best	121	485	898.80	150.0	0.341	826.28	1.32x10 ⁻⁹
		Worst	121	485	898.80	150.0	0.341	826.28	
		Average						826.28	
Varied	10000	Best	121	485	898.80	150.0	0.341	826.28	157.46
		Worst	279	408	1507.53	216.1	0.296	1188.98	
		Average						964.04	
Constant	10000	Best	121	485	898.80	150.0	0.341	826.28	85.84
		Worst	187	496	1231.94	214.5	0.298	1040.78	
		Average						826.28	

Table 8 Results from Testing a Variable Dimension Penalty

5 Conclusions

The PSO has proved to be a relatively robust tool for exploring optimal solutions for reinforced concrete beams. It has not been very sensitive to the variations in the PSO parameters for swarm size, acceleration and momentum. The tests all found a similar solution. When the penalty functions were examined, it was found that fine tuning was necessary. Inappropriate values of the penalty functions can lead to a breakdown in the search.

The tests conducted investigated the effect of different parameters of the objective function and the PSO algorithm on the performance and results of the developed PSO program. The results indicated that the swarm size, acceleration coefficients and momentum coefficients have an effect on the processing time and convergence behaviour of the PSO algorithm. The value penalty coefficients had a significant effect on the results, causing non-convergence when very large penalty coefficients were used, and convergence on non-viable results when very small penalty coefficients were used.

One interesting result was the effect of linearly increasing the dimension ratio penalty function throughout the algorithm process. Linearly increasing the penalty coefficient had a tendency to cause the particles to converge on an initial position, not covering the search space and hence not finding the optimum solution. By using a constant dimension ratio penalty, the results converged more consistently.

References

- [1] S. Rajeev and C. S. Krishnamoorthy, "Genetic algorithms-based methodologies for design optimization of trusses," *Journal of Structural Engineering*, pp. 350-358, 1997.
- [2] T J McCarthy and A Fenwick, "Truss topology optimisation using genetic algorithms", Paper AI.9, 9th Int Conf Applications of Artificial Intelligence in Civil, Structural and Environmental Engineering, BHV Topping (Ed) Malta, 18-21 Sept 2007, ISBN 978-1-905088-20-1
- [3] Abraham, A., C. Grosan, and V. Ramos, *Swarm intelligence in data mining*. 2006, Berlin: Springer.
- [4] Engelbrecht, A.P., *Fundamentals of computational swarm intelligence*. 2005, Hoboken, New Jersey, Wiley.
- [5] Brits, R., A.P. Engelbrecht, and F. Van den Bergh. *A Niching Particle Swarm Optimisation*. in *Fourth Asia-Pacific Conference on Simulated Evolution and Learning*. 2002.
- [6] Van den Bergh, F. and A.P. Engelbrecht. *Effects of Swarm Size on Cooperative Particle Swarm Optimisers*. in *Genetic and Evolutionary Computation Conference*. 2001.
- [7] Ratnaweera, A.C., S.K. Halgamuge, and H.C. Watson. *Particle Swarm Optimiser with Time Varying Acceleration Coefficients*. in *International Conference on Soft Computing in Intelligent Systems*. 2002.
- [8] Al-Salloum, Y.A. and G.H. Siddiqi, *Cost-optimum design of reinforced concrete beams*. ACI Structural Journal, 1994. **91**(6): p. 647-655.
- [9] Leps, M. and M. Sejnoha, *New approach to optimization of reinforced concrete beams*. Computers and Structures, 2003. **81**(18-19): p. 1957-1966.
- [10] Standards Association of Australia, *Concrete structures : AS 3600-2001*. 8th ed. 2001, Homebush, N.S.W. Standards Australia International
- [11] Warner, R.F., S.J. Foster, and A.E. Kilpatrick, *Reinforced concrete basics : analysis and design of reinforced concrete structures*. 2007, Frenchs Forest, N.S.W. :: Pearson Prentice Hall.
- [12] Warner, R.F., *Concrete structures* 1998, South Melbourne, Longman
- [13] *Rawlinson's Australian construction handbook*. 1983, [Perth, W.A.] Rawlhouse Publishing.