COS 710
Assignment 1

Jordan Daubinet, u15260870

March 2018

1 Introduction

The particle swarm optimization algorithm (PSO)is commonly used for solving multi-dimensional optimization problems. It does not guarantee finding a global optimal but a local optimal which may be the global optimal.

Many studies (quote studies) have shown that a standard PSO is not feasible for producing accurate and consistent results due to its roaming behaviour and the effect velocity has on its roaming behaviour, producing unfeasible solutions.

I will be implementing and analyzing the results of 11 different boundary constraint handling mechanisms (quote the boundary handling mechanisms) which influence the way the velocity and/or particle positions are updated in an attempt to constrain velocities influence on particle positions and/or change velocities influence to produce more feasible solutions.

2 Background

2.1 Problem Domain

The issues of the PSO roaming behaviour stem from the way it updates individual particle positions and velocities, in particular the effect velocity has on updating particle positions.

Without implementing some boundary constraint handling mechanisms, the velocities effect on the algorithm cause particle's positions to leave the boundaries of the feasible space throughout all dimensions which produces an unfeasible solution to the problem.

2.2 Basic Particle Swarm Optimization

The main components of a PSO are the following:

- **a swarm of particles** which together, during the algorithms execution, explore and exploit the search space and eventually converge to a position in the search space which they find optimal.
- **each particle represents a candidate solution** which has its own position in the search space as well as a velocity, two positive acceleration coefficients an randomization element which guide it through the search space.
- **positions of a particle represent parameters to be optimized**, during the algorithms execution, these positions are searched for optimal values.

For this assignment [quote assignment paper] I implemented a global PSO which uses memory to guide the global best, where the global best position is selected as the best particles personal best position and particles are updated synchronously. The basic structure of a gbest PSO are described below:

Position initialization: let $x_{ij}(0)$ represent a position value at dimension i for particle j at time 0, then every position in each particle is initialized by:

$$x_{ij}(0)\ U(x_{minj}, x_{max,j})$$

Position updates: let x_i be a position value at dimension i and vi be a velocity value at dimension i for a particle in the swarm, where t represents time. Then a position is updated by:

$$xi(t+1) = xi(t) + vi(t+1)$$

Velocity initialization: let $x_{ij}(0)$ represent a velocity value at dimension i for particle j at time 0, then every velocity in each particle is initialized by:

$$x_{ij}(0) = 0$$

Velocity update per dimension: let v_{ij} be a velocity value for particle i at dimension j and $y_{ij}(t)$ be the personal best position of particle i at dimension j at time t and $\hat{y}_{ij}(t)$be the global best position of particle i at dimension j at time t. Then velocity is updated by:

$$v_{ij}(t+1) = w * v_{ij}(t) + c_1 r_1 j(t)[y_{ij}(t) - x_{ij}(t)] + c2r2j(t)[\hat{y}_{ij}(t) - x)_{ij}(t)]$$
$$c_1 = c_2 = 1.49618,$$
$$w = 0.729844, \qquad r_1 = r_2 = U(0,1)$$

Personal Best:

$$y_i(t+1) = \{\, y_{\,i}(t) \qquad if \quad f(x_i(t+1)) \geq f(y_i(t))$$

$$\{ y_i(t+1) \quad if \quad f(x_i(t+1)) < f(y_i(t))$$

Global Best:

$\hat{y}(t) = min\{f(x_0(t), ..., f(x_{n_s}(t))\}$
where n_s is the number of particles in the swarm.

Pseudo code of Algorithm:

```
repeat
    for each particle i = 1, ..., S.n_s do
        if f(S.x_i) < f(S.y_i) then
            S.y_i = S.x_i
        end
        if f(S.y_i) < f(S.ŷ_i) then
            S.y_i = S.x_i
        end
    end
    for each particle i = 1, ..., S.n_s do
        update the velocity
        update the position
    end
until stopping condition is true
```

2.3 Boundary constraint handling mechanisms

The boundary constraint handling mechanisms which I implemented are described below:

2.3.1 Feasible positions : c_1

Update the personal best positions only if the new particle position is better than its current personal best, and if the new particle position is feasible. That is, a new particle position can not become a personal best position if it violates boundary constraints.

2.3.2 Clamping approach : c_2

If a particle violates a boundary constraint in a specific dimension, then clamp the corresponding decision variable at the boundary value.

2.3.3 Per element re initialization : c_3

For any decision variable of any particle that violates a boundary constraint, reinitialize that decision variable to a random position that satisfies the boundary constraints.

2.3.4 Per element re initialization and setting velocity to zero : c_4

Adapt the per element re initialization approach above to also set the velocity of the decision variable that violates a boundary constraint to zero. The corresponding decision variable's new position will therefore not be influenced by the momentum term.

2.3.5 Initialize to personal best position : c_5

Initialize the boundary violating decision variable to the corresponding personal best position.

2.3.6 Initialize to personal best position and set velocity to zero : c_6

Adapt the initialize to personal best position strategy c_5 to also set the corresponding velocity to 0.

2.3.7 Initialize to global best position : c_7

Adapt strategy c_5 to Initialize the boundary violating decision variable to the corresponding global best position.

2.3.8 Initialize to global best position and set velocity to zero : c_8

Adapt strategy c_7 to also set the corresponding velocity to 0.

2.3.9 Reverse velocity : c_9

The velocity of the boundary violating decision variable is simply reversed while that decision variable violates the boundary constraint.

2.3.10 Use an arithmetic average : c_{10}

Set the boundary violating decision variable to an arithmetic average of the corresponding personal best and global best position

2.3.11 Reverse velocity version 2: c_{11}

Only update the velocity when in all positions are within the bounds and only reverse the velocity once when it leaves the bounds, therefore causing it to not grow when out of bounds and guide the search back to within the bounds

2.3.12 Alternative Boundary constraint handling mechanisms

[1][2][3]

3 Implementation

3.1 control parameters of PSO

3.1.1 Algorithm processes and parameters

For all performance measures [quote tables], Each run consisted of one benchmark function [quote] and one boundary constraint [quote]. I used a swarm of 30 particles which executed for 5000 iterations in one run. I repeated each run 50 times to get an average result for a benchmark Function/boundary constraint pair. For each function [quote benchmark functions] this process was repeated with every boundary constraint for a total of 11 times.

3.1.2 Performance measure: Accuracy

To produce my accuracy results [quote table] I ran the above stated algorithm process with the specified parameters, testing each function with 30 dimensions.

3.1.3 Performance measure: Efficiency

To produce my efficiency results, I measured the average time it took for a run to reach a certain accuracy level. I set different accuracy levels per function, depending on reachable values which where in the the functions domain [quote table].

3.1.4 Performance measure: Scalability

To produce my scalability results, I measured the accuracy of each function [quote] against each constraint [quote] with both 30 dimensions [quote table] and 5 dimensions [quote table] and produced a table of the average accuracy between these two result sets.

3.1.5 Performance measure: least amount of search effort outside of the feasible region

To produce a measure for which boundary constraint wasted the least amount of time, I averaged over all particles in my swarm the duration that a particle was out of the bounds of any dimension to the time that, that same particle was in the bounds of all dimensions. Resulting in percentage of time [quote table].

3.2 Benchmark Functions

The first 9 [quote functions] functions I used where taken from article [quote], where by shifting parameters are specified. for my last function [quote function] was taken from [quote the article] whereby parameters are given for shifting and rotating. for all functions which used rotation, I generated a random orthonormal matrix and used this same matrix with every call to the function, that is every time a particle called the function for every 1 of the

5000 iterations as well as each of the 50 runs of the pso.

My choice of functions were selected based on unique characteristics of the Unimodal and Multimodal function classes. Therefore For both Unimodal and Multimodal, I selected functions which had one of the following characteristics: Separable, Non-separable, Noisy, Shifted, Rotated. This resulted in the 10 functions specified below.

3.2.1 the absolute value function : f_1

$f_1 = \sum_{j=1}^{n_x} |x_j|, \qquad x_j \epsilon [-100, 100]$

3.2.2 the schwefel 1.2 function : f_2

$f_2 = \sum_{j=1}^{n_x} \left(\sum_{i=1}^{i} x_i \right)^2, \qquad x_j \epsilon [-100, 100]$

3.2.3 de jong's f4 function : f_3

$f_3 = \sum_{j=1}^{n_x} (jx^4 + N(0,1), \qquad x_j \epsilon [-1.28, 1.28]$

3.2.4 the elliptic function : f_4

$f_3 = \sum_{j=1}^{n_x} (10^6)^{\frac{j-1}{n_x - 1}}, \qquad x_j \epsilon [-100, 100]$

3.2.5 the schwefel 1.2 function, rotated : f_5

$f_2 = \sum_{j=1}^{n_x} \left(\sum_{i=1}^{i} x_i \right)^2, \qquad x_j \epsilon [-100, 100]$

3.2.6 the alpine function : f_6

$(\prod_{j=1}^{n_x} sin(x_j)) \sqrt{\prod_{j=1}^{n_x} x_j}, \qquad x_j \epsilon [-10, 10]$

3.2.7 the egg holder function : f_7

$f_7 = \sum_{j=1}^{n_x - 1} (-(x_{j+1}+47) sin(\sqrt{\left| x_{j+1} + \frac{x_j}{2} + 47 \right|}) + sin(\sqrt{|x_j - (xj + 1 + 47)|})(-x_j)),$
$x_j \epsilon [-512, 512]$

3.2.8 the griewank function : f_8

$1 + \frac{1}{4000} \sum_{j=1}^{n_x} x_j^2 - \prod_{j=1}^{n_x} cos(\frac{x_j}{\sqrt{j}}), \qquad x_j \epsilon [-600, 600]$

3.2.9 the rosenbrock function : f_9

$\sum_{j=1}^{n_x - 1} (100(x_{j+1} - x_j^2)^2 + (x_j - 1)^2), \qquad x_j \epsilon [-30, 30]$

3.2.10 the Rastrigin's function, rotated CEC f:17 : f_{10}

$\sum_{j=1}^{n_x}(x_j^2 - 10cos(2\pi x_j) + 10), \qquad x_j \epsilon [-5, 5]$

3.3 constraints

Whenever a particle becomes unfeasable, that is its positions go out of a boundary, a boundary constraint mechanism is used in an attempt to bring that particle back into the feasible search space. The boundary constraint mechanism acts upon the velocity v_{ij} and/or position x_{ij} of the respective dimension which has violated the bounds.

for all constraint handling mechanisms below, let $x_{ij}(t)$ particle i at position j at time t.

3.3.1 Feasible positions : c_1

if $x_{ij} > Upper_{boundsj}$ *or* $x_{ij} < Lower_{boundsj}$, and
$x_i(t+1) < y_i(t)$, then
$x_i(t+1) = x_i(t)$

3.3.2 Clamping approach : c_2

if $x_{ij} > Upper_{boundsj}$, then
$x_i(t+1) = Upper_{boundsj}$, else
if $x_{ij} < Lower_{boundsj}$, then

$x_i(t+1) = Lower_{boundsj}$

3.3.3 Per element re initialization : c_3

if $x_{ij} > Upper_{boundsj}$ *or* $x_{ij} < Lower_{boundsj}$, then
$x_{ij}(t+1) = U(x_{minj}, x_{max,j})$

3.3.4 Per element re initialization and setting velocity to zero: c_4

if $x_{ij} > Upper_{boundsj}$ *or* $x_{ij} < Lower_{boundsj}$, then
$x_{ij}(t+1) = U(x_{minj}, x_{max,j})$, and
$v_{ij}(t+1) = 0$

3.3.5 Initialize to personal best position: c_5

if $x_{ij} > Upper_{boundsj}$ *or* $x_{ij} < Lower_{boundsj}$, then
$x_{ij}(t+1) = y_{ij}(t)$

3.3.6 Initialize to personal best position and set velocity to zero: c_6

if $x_{ij} > Upper_{boundsj}$ *or* $x_{ij} < Lower_{boundsj}$, then
$x_{ij}(t+1) = y_{ij}(t)$, and
$v_{ij}(t+1) = 0$

3.3.7 Initialize to global best position: c_7

if $x_{ij} > Upper_{boundsj}$ *or* $x_{ij} < Lower_{boundsj}$, then
$x_{ij}(t+1) = \hat{y}_{ij}(t)$

3.3.8 Initialize to global best position and set velocity to zero: c_8

if $x_{ij} > Upper_{boundsj}$ *or* $x_{ij} < Lower_{boundsj}$, then
$x_{ij}(t+1) = \hat{y}_{ij}(t)$, and
$v_{ij}(t+1) = 0$

3.3.9 Reverse velocity: c_9

if $x_{ij} > Upper_{boundsj}$ *or* $x_{ij} < Lower_{boundsj}$, then
$v_{ij}(t+1) = w * -v_{ij}(t) + c_1 r_1 j(t)[y_{ij}(t) - x_{ij}(t)] + c2r2j(t)[\hat{y}_{ij}(t) - x)_{ij}(t)]$
$c_1 = c_2 = 1.49618$,
$w = 0.729844$, $r_1 = r_2 = U(0,1)$

3.3.10 Use an arithmetic average: c_{10}

if $x_{ij} > Upper_{boundsj}$ *or* $x_{ij} < Lower_{boundsj}$, then
$x_{ij}(t+1) = \alpha y_{ij}(t) + (1-\alpha)\hat{y}_{ij}$, where
α $U(0,1)$

3.3.11 Reverse velocity version 2e: c_{11}

if $x_{ij} > Upper_{boundsj}$ *or* $x_{ij} < Lower_{boundsj}$, and
this is the first iteration that the particle
has gone out of bounds, then
$v_{ij}(t+1) = -v_{ij}(t)$

4 Research Results

4.1 Performance test 1

Performance test 1 was used When measuring performance with relation to solution accuracy. Performance for this test was based on which constraint leaded to the lowest value within the bounds, therefore our PSO is a minimization optimizer.The results [Table 1][Table 2][Table 3] show that both $c4$ and $c7$ had the best results over the functions in 30% of the tests and we could conclude that these constraints mechanism produced the most accurate results for our PSO. The results also show that $c6$ had the lowest standard deviation over the functions in 30% of the tests and we could conclude that this constraint mechanism produces the most consistent results.

We can also observer that throughout the performance test, both constraints which deal with reverse velocity of some sort produce poor performance in both accuracy and consistency. Well it is expected of c11, due to its ever increasing nature of its velocity, it was not expected of c9 as its velocity does not continuously increase. however it can be concluded that since its velocity value never decreases either, it only reverses its direction, that this is the cause to its poor performance as it is common in both c9 and c11.

Table 1: 50 dimensions, 5000 iterations, 50 runs

	absolute value (f1)	schwefel 1.2 (f2)	dejong's f4 (f3)
c1	4.23e-09 +- 2.951e-08	8.85e-09 +- 1.48e-08	-16.11 +- 4.31
c2	70.00 +- 72.80	8233.33 +- 7662.39	-19.65 +- 2.27
c3	5.60e-12 +- 3.21e-11	7.18e-09 +- 1.38e-08	-19.44 +- 2.13
c4	3.01e-10 +- 2.05e-09	3.94e-09 +- 4.49e-09	-19.44 +- 2.13
c5	4.52e-09 +- 2.50e-08	1.18e-08 +- 3.05e-08	-19.04 +- 1.99
c6	3.35e-10 +- 1.87e-09	8.98e-09 +- 1.59e-08	-17.67 +- 2.44
c7	4.51e-09 +- 3.15e-08	101.64 +- 699.85	-16.64 +- 3.17
c8	3.05e-09 +- 1.64e-08	1.72e-08 +- 4.52e-08	-15.98 +- 3.81
c11	3.24 +- 12.79	51111.59 +- 23005.23	-5.72 +- 9.14
c10	1.07e-09 +- 7.45e-09	1.70e-08 +- 7.60e-08	-17.63 +- 2.32
c9	889.36 +- 118.72	10.17e+04 +- 22.30e-03	53.45 +- 14.50

Table 2: 50 dimensions, 5000 iterations, 50 runs

	elliptic (f4)	schwefel 1.2: rotated (f5)	alpine (f6)
c1	-11.43e+07 +- 992476.01	1.05e-08 +- 1.26e-08	-15.86e+10 +- 2.01e+11
c2	-99.79e+06 +- 7258108.75	7.69e+03 +- 7.30e+03	-65.24e+09 +- 1.23e+11
c3	-65.16e+06 +- 27.16e+05	1.56e-08 +- 2.41e-08	-71.97e+10 +- 4.73e+11
c4	-10.55e+07 +- 22.05+e05	5.19e-09 +- 9.52e-09	-39.85e+10 +- 5.46e+11
c5	-10.99e+07 +- 21.46+05	3.46e-09 +- 4.48e-09	-8.31e+11 +- 5.20e+11
c6	-11.48e+07 +- 1.53e-08	7.23e-09 +- 1.15e-08	-1.76e+11 +- 3.74e+10
c7	-11.48e+07 +- 1.72e-08	4.62e+02 +- 1.38+03	-5.53e+10 +- 5.88e+10
c8	-11.36e+07 +- 15.96e+05	3.74e-09 +- 4.34e-09	-6.81e+11 +- 7.96e+10
c11	-inf +- nan	4.99e+04 +- 3.41e+04	-inf +- nan
c10	-11.48e+07 +- 1.61e-08	1.98e-08 +- 4.58e-08	-1.08e+11 +- 1.42e+12
c9	-inf +- nan	7.71e+04 +- 1.79e-04	-1.58e+83 +- 4.62e+83

Table 3: 50 dimensions, 5000 iterations, 50 runs

	egg holder (f7)	griewank (f8)	rosenbrock (f9)	Rastrigin: rotated (f10)
c1	-1.31e+04 +- 1.26e+03	-179 +- 2.10e-13	51.07 +- 42.25	4.40e-03 +- 4.52e-03
c2	-1.50e+04. +- 1.045e+03	-175.40 +- 17.63	7.13e+04 +- 1.52e+05	3.87e-03 +- 2.73e-03
c3	-8.47e+03 +- 1.23e+03	-179 +- 7.45e-14	26.67 +- 20.09	1.72e-03 +-1.33e-03
c4	-1.28e+04 +- 1.24e+03	-179 +- 6.18e-14	58.32 +- 78.05	3.79e-03 +- 2.96e-03
c5	-1.39e+04 +- 1.31e+03	-179 +- 3.24e-14	27.02 +- 23.13	2.98e-03 +- 3.61e-03
c6	-1.45e+04 +- 1.53e+03	-179 +- 3.08e-14	32.66 +- 21.44	3.94e-03 +- 1.91e-03
c7	-1.59e+04 +- 1.16e+03	-179 +- 7.70e-13	793.29 +- 2164.19	5.00e-03 +- 4.93e-03
c8	-1.49e+04 +- 1.50e+03	-179 +- 1.07e-13	116.78 +- 197.71	5.63e-03 +- 6.55e-03
c11	-inf +- nan	-175.87 +- 16.21	2.85e+06 +- 6.68e+06	8.67e-02 +- 7.40e-03
c10	-1.53e+04 +- 1.54e+03	-179 +- 2.14e-13	162.00 +- 215.24	1.27e-03 +- 1.12e-03
c9	-inf +- nan	172.46 +- 60.14186	1.28e+08 +- 5.78e+07	1.78e-01 +- 2.30e-01

4.2 Performance test 2

Performance test 2 was used When measuring performance with relation to scaling. Performance for this test was based on which constraint leaded to the lowest value within the bounds when only using 5 dimensions for our search space, PSO is a minimization optimizer.The results [Table 4][Table 5][Table 6] show that *c*5had the best results over the functions in 40% of the tests and we could conclude that this constraints mechanism produced the most accurate results for our PSO when only using 5 dimensions.

Table 4: 5 dimensions, 5000 iterations, 50 runs

	absolute value (f1)	schwefel 1.2 (f16)	dejong's f4 (f11)
c1	4.488e-170 +- 0.0	37.407 +- 8.828	690.630 +- 238.724
c2	5.095e-169 +- 0.0	3.045e-260 +- 0.0	-9.569 +- 0.549
c3	2.188e-169 +- 0.0	4.963e-261 +- 0.0	-9.583 +- 0.654
c4	7.634e-170 +- 0.0	1.851e-261 +- 0.0	-9.653 +- 0.529
c5	3.322e-169 +- 0.0	4.9147e-261 +- 0.0	-10.2082 +- 0.738
c6	8.193e-170 +- 0.0	6.216e-259 +- 0.0	-9.753 +- 0.624
c7	6.107e-169 +- 0.0	3.107e-259 +- 0.0	-9.764 +- 0.775
c8	3.024e-170 +- 0.0	5.357e-263 +- 0.0	-9.777 +- 0.655
c11	6.418e-168 +- 0.0	8.186e-255 +- 0.0	-9.254 +- 0.531
c10	2.120e-170 +- 0.0	2.070e-260 +- 0.0	-10.160 +- 0.851
c9	37.407 +- 8.828	690.630+- 238.724	-5.898 +- 0.613

Table 5: 5 dimensions, 5000 iterations, 50 runs

	elliptic (f5)	schwefel 1.2: rotated (f16)	alpine (f3)
c1	-13.87e+07 +- 0.0	2.484e-30 +- 3.090e-30	-130.358 +- 16.117
c2	-13.87e+07 +- 0.0	8.552e-30 +- 5.564e-30	-128.478 +- 21.758
c3	-12.85e+07 +- 25.95e+05	3.312e-11 +- 9.935e-11	-135.731 +- 2.842e-14
c4	-13.81e+07 +- 20.35e+04	2.101e-30 +- 2.584e-30	-130.358 +- 16.117
c5	-13.87e+07 +- 1.138	5.592e-30 +- 6.407e-30	-130.359 +- 16.117
c6	-13.87e+07 +- 0.0	6.387e-30 +- 5.141e-30	-117.733 +- 27.917
c7	-13.87e+07 +- 0.0	4.839e-30 +- 3.562e-30	-114.488 +- 33.258
c8	-13.79e+07 +- 79.30e+04	1.104e-29 +- 1.410e-29	-130.358+- 16.117
c11	-inf +- nan	9.163e-30 +- 8.476e-30	-1.801e+41 +- 2.809e+41
c10	-13.87e+07 +- 0.0	8.453e-30 +- 8.116e-30	-128.478 +- 21.758
c9	-inf +- nan	110.146 +- 86.365	-3.999e+118 +- 1.120e+119

Table 6: 5 dimensions, 5000 iterations, 50 runs

	egg holder f4)	griewank (f6)	rosenbrock (f13)	Rastrigin: rotated (f29)
c1	-2994.608 +- 351.946	-179.0 +- 0.0	4.472 +- 7.224	5.09e-04 +- 3.55e-04
c2	-2856.873 +- 332.712	-179.0 +- 0.0	19.53e+03 +- 58.55e+03	5.60e=04 +- 4.45e-04
c3	-2784.287 +- 210.011	-179.0 +- 0.0	1.434 +- 2.216	2.55e-04 +- 2.19e-04
c4	-2923.312 +- 286.259	-179.0 +- 0.0	7.525 +- 9.664	3.09e-04 +- 2.46e-04
c5	-3397.516 +- 291.258	-179.0 +- 0.0	1.887 +- 4.405	5.37e-04 +- 2.77e-04
c6	-3151.772 +- 321.509	-179.0 +- 0.0	0.331 +- 0.990	3.78e-04 +- 2.20e-04
c7	-2776.708 +- 338.609	-179.0 +- 0.0	5.976 +- 8.829	4.61e-04+- 2.92e-04
c8	-2713.075 +- 172.802	-179.0 +- 0.0	7.084 +- 11.536	4.73e-04 +- 4.42e-04
c11	-inf +- nan	-179.0 +- 0.0	5.284+- 6.3851	2.20e-03 +- 2.17e-03
c10	-2923.2336 +- 458.710	179.0 +- 0.0	5.793 +- 7.695	5.73e-04 +- 7.13e-04
c9	-inf +- nan	-173.943 +- 2.703	21.34e+03 +- 14.05e+03	3.12e-03 +- 2.64e-04

4.3 Performance test 3

Performance test 3 was used When measuring performance with relation to efficiency. Performance for this test was based on which constraint reached the required accuracy level value within the shortest amount of time as a percentage, PSO is a minimization optimizer. The results [Table 7][Table 8][Table 9] show that $c4$ had the best results over the functions in 20% of the tests and we could conclude that this constraints mechanism produced the most efficient results for our PSO.

Table 7: The average time to get to specified accuracy level

	absolute value (40%)	schwefel 1.2 (40%)	dejong's f4 (90%)
c1	0.0001861821793	2.885	0.142
c2	0.259	2.981	0.367
c3	0.354	2.436	0.173
c4	0.219	2.729	0.084
c5	0.252	3.017	0.161
c6	0.274	2.364	0.043
c7	0.429	2.12	0.137
c8	0.767	2.365	0.135
c11	3.485	NAN	0.34
c10	0.774	2.449	0.156
c9	NAN	NAN	NAN

Table 8: The average time to get to specified accuracy level

	elliptic (90%)	schwefel 1.2: rotated (40%)	alpine (90%)
c1	4.41E-05	12.108	3.70E-05
c2	4.09E-05	2.116	4.44E-05
c3	4.20E-05	6.753	5.04E-05
c4	4.06E-05	6.057	0.0002408379326
c5	3.91E-05	6.268	5.01E-05
c6	7.69E-05	6.266	8.53E-05
c7	3.70E-05	7.823	5.25E-05
c8	3.91E-05	6.8	5.11E-05
c11	3.95E-05	NAN	5.43E-05
c10	3.60E-05	6.287	5.32E-05
c9	3.70E-05	NAN	5.43E-05

Table 9: The average time to get to specified accuracy level

	egg holder (90%)	griewank (60%)	rosenbrock (20%)	Rastrigin: rotated (40%)
c1	0.0001935871523	0.099	6.039	0.814
c2	0.0001741931753	0.095	3.117	0.581
c3	0.000179129824	0.098	12.126	0.648
c4	0.001002492302	0.086	6.527	0.201
c5	0.0001893557391	0.099	5.129	0.292
c6	0.0002080444806	0.089	5.231	0.272
c7	0.00047462351	0.168	4.851	0.208
c8	0.0003265240492	0.105	3.075	0.312
c11	0.0008124313277	0.171	0.402	0.955
c10	0.0002771575623	0.092	3.736	0.364
c9	0.0001861821793	0.092	NAN	12.108

4.4 Performance test 4

Performance test 4 was used When measuring performance with relation to search effort. Performance for this test was based on which constraint remained within the search space for the larges amount of time as a percentage, PSO is a minimization optimizer. The results [Table 10][Table 11][Table 12] show that many of the constraints remained within the bounds for the duration of the PSO's execution, the only poor performing constraints which are measurable are $c9$ and $c11$, which is to be expected as these constraints constantly increase the velocity which causes our positions to leave the search space.

Table 10: The average time that the particles were in a feasible search space as a percentage of total run time

	absolute value (f1)	schwefel 1.2 (f16)	dejong's f4 (f11)
c1	0.989	0.976	0.919
c2	1	1	1
c3	1	1	1
c4	1	1	1
c5	1	1	1
c6	1	1	1
c7	1	1	1
c8	1	1	1
c11	0.989	0.067	0.04
c10	1	1	1
c9	0	0	0

Table 11: The average time that the particles were in a feasible search space as a percentage of total run time

	elliptic (f5)	schwefel 1.2: rotated (f16)	alpine (f3)
c1	0	0.978	0
c2	1	1	1
c3	1	1	1
c4	1	1	1
c5	1	1	1
c6	1	1	1
c7	1	1	1
c8	1	1	1
c11	0	0.978	0
c10	1	1	1
c9	0	0	0

Table 12: The average time that the particles were in a feasible search space as a percentage of total run time

	egg holder f4)	griewank (f6)	rosenbrock (f13)	Rastrigin: rotated (f29)
c1	0	0.989	0	0.994
c2	1	1	1	1
c3	1	1	1	1
c4	1	1	1	1
c5	1	1	1	1
c6	1	1	1	1
c7	1	1	1	1
c8	1	1	1	1
c11	0	0.363	0.265	0.265
c10	1	1	1	1
c9	0	0	0	0

5 Conclusion

Throughout the assignment i have learned one distinct thing, no constraint is perfect for all problems and in our situation, all benchmark functions. Some perform faster, others more accurate and others scale better for different dimensions. In my experiment this would be $c4$, $c7$ and $c4$ respectfully, although we can clearly see that the measures by which these constraints perform better than the others is minimal, 40% at best and therefore conclude that no constraint mechanism is much better than any other. However many perform allot better than the worst, the worst being versions of reverse velocity.
It has been interesting investigating how simple changes to the way the positions or the velocities of the PSO, change the behaviour and therefore effect many aspects of it.

6 references

[1]A Constraint-Handling Mechanism for Particle Swarm Optimization. (2004). [ebook] Mexico. Available at: https://pdfs.semanticscholar.org/be9a/76f0d07c4dc137a7e2f1b9d0f857b56ea
[Accessed 4 Mar. 2018].

[2]A Constraint-Handling Mechanism for Particle Swarm Optimization. (n.d.). Mexico: Gregorio Toscano Pulido and Carlos A. Coello Coello.

[3]Boundary Handling Approaches in Particle Swarm Optimization. (2012). [ebook] Cambridge: Nikhil Padhye. Available at: http://www.egr.msu.edu/ kdeb/papers/k2012014.pdf
[Accessed 4 Mar. 2018].