A New Particle Swarm Optimiser for Linearly Constrained Optimisation

Ulrich Paquet
Department of Computer Science
University of Pretoria
South Africa
upaquet@cs.up.ac.za

Andries P. Engelbrecht
Department of Computer Science
University of Pretoria
South Africa
engel@driesie.cs.up.ac.za

Abstract- A new PSO algorithm, the Linear PSO (LPSO), is developed to optimise functions constrained by linear constraints of the form $A\mathbf{x} = \mathbf{b}$. A crucial property of the LPSO is that the possible movement of particles through vector spaces is guaranteed by the velocity and position update equations. This property makes the LPSO ideal in optimising linearly constrained problems. The LPSO is extended to the Converging Linear PSO, which is guaranteed to always find at least a local minimum.

1 Introduction

Particle Swarm Optimisation (PSO) was originally introduced by Kennedy and Eberhart [5], and has proved to be a very useful algorithm to optimise unconstrained functions. If a number of constraints is added to the objective function (the function that is optimised), the problem becomes more complicated. A number of approaches has been taken in the Evolutionary Computing field to do constraint handling. The three main approaches are penalty, repair, and constraint-preserving methods.

Penalty methods add a penalty to the objective function to decrease the quality of infeasible solutions [3, 4, 11]. While penalty methods are very popular, they do not guarantee a solution where no constraints are violated, since the search space still includes infeasible solutions, and success depends on the penalty method.

Repair methods apply operators to move an infeasible solution closer to the feasible space of solutions [9, 17]. Operators designed to 'correct' infeasible solutions are usually computationally intensive. Not all constraints can be easily implemented to be corrected by these operators, which must be tailored to the particular problem [1].

Constraint-preserving methods (feasible solutions methods) reduce the search space by ensuring that all candidate solutions at all times satisfy the constraints [11]. Solutions are initialised within the feasible domain, and transformations of candidate solutions are such that the resulting solutions still lie within the feasible domain.

A new PSO algorithm, the Linear PSO (together with the Converging Linear PSO), is introduced to optimise problems with linear equality constraints. The new algorithm is a *constraint-preserving* method. It adds no extra cost of implementation, except for initialising all particles to be feasible solutions. Experimental results indicate its efficiency and ease of implementation as an optimiser for linearly constrained problems.

The rest of this paper is organised as follows: Section 2 provides an overview of PSO, while Section 3 extends the standard PSO to an algorithm suited for constrained optimisation. Finally, Section 4 shows experimental results to substantiate the new algorithm's success.

2 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO) was originally introduced by Kennedy and Eberhart [5], and has its roots in swarm intelligence. The motivation behind the algorithm is the intelligent collective behaviour of organisms in a swarm (e.g. a flock of birds migrating), while the behaviour of a single organism in the swarm may seem totally inefficient.

PSO represents an optimisation method where particles collaborate as a population to reach a collective goal. Each n-dimensional particle $\mathbf{x}_i$ is a potential solution to the collective goal, usually to minimise a function f. PSO differs from traditional optimisation methods, in that a population of potential solutions are used in the search. The direct fitness information instead of function derivatives or other related knowledge is used to guide the search. This search is based on probabilistic, rather than deterministic, transition rules.

A particle $\mathbf{x}_i$ has memory of the best solution $\mathbf{y}_i$ that it has found, called its *personal best*; it flies through the search space with a velocity $\mathbf{v}_i$, which is dynamically adjusted according to its personal best and the *global best* solution $\widehat{\mathbf{y}}$ found by the rest of the swarm (called the *gbest* topology). Other topologies for information sharing have also been investigated [6, 7, 8].

Let i indicate a particle's index in the swarm, such that $\mathcal{S} = \{\mathbf{x}_1, \ldots, \mathbf{x}_s\}$ is a swarm of s particles. During each iteration of the PSO algorithm, the personal best $\mathbf{y}_i$ of each particle is compared to its current performance, and set to the better performance. If the objective function to be min-

imised is defined as $f : \mathbb{R}^n \rightarrow \mathbb{R}$, then

$$\mathbf{y}_i^{(t)} = \begin{cases} \mathbf{y}_i^{(t-1)} & \text{if } f(\mathbf{x}_i^{(t)}) \geq f(\mathbf{y}_i^{(t-1)}) \\ \mathbf{x}_i^{(t)} & \text{if } f(\mathbf{x}_i^{(t)}) < f(\mathbf{y}_i^{(t-1)}) \end{cases} \quad (1)$$

The global best $\widehat{\mathbf{y}}$ is updated to the position with the best performance within the swarm, with

$$\begin{aligned} \widehat{\mathbf{y}}^{(t)} &\in \{\mathbf{y}_1^{(t)}, \mathbf{y}_2^{(t)}, \ldots, \mathbf{y}_s^{(t)}\} \mid f(\widehat{\mathbf{y}}^{(t)}) \\ &= \min\{f(\mathbf{y}_1^{(t)}), f(\mathbf{y}_2^{(t)}), \ldots, f(\mathbf{y}_s^{(t)})\} \end{aligned} \quad (2)$$

Traditionally, each particle's velocity and position is updated separately for each dimension j, with

$$\begin{aligned} v_{ij}^{(t+1)} &= w v_{ij}^{(t)} + c_1 r_{1j}^{(t)}[y_{ij}^{(t)} - x_{ij}^{(t)}] \\ &\quad + c_2 r_{2j}^{(t)}[\widehat{y}_j^{(t)} - x_{ij}^{(t)}] \end{aligned} \quad (3)$$

$$x_{ij}^{(t+1)} = v_{ij}^{(t+1)} + x_{ij}^{(t)} \quad (4)$$

The stochastic nature of the algorithm is determined by $r_{1j}^{(t)}$ and $r_{2j}^{(t)}$, two uniform random numbers between zero and one. These random numbers are scaled by acceleration coefficients c_1 and c_2, where $0 \leq c_1, c_2 \leq 2$. The inertia weight w was introduced to improve the convergence rate of the PSO algorithm [14]. It is possible to clamp the velocity vectors by specifying upper and lower bounds on $\mathbf{v}_i$, to avoid too rapid movement of particles in the search space.

The standard PSO algorithm is summarised below:

Algorithm 1 – Standard Particle Swarm Optimiser

1. Set the iteration number t to zero, and randomly initialise swarm $\mathcal{S}$ within the search space.
2. Evaluate the performance $f(\mathbf{x}_i^{(t)})$ of each particle.
3. Compare the personal best of each particle to its current performance, and set $\mathbf{y}_i^{(t)}$ to the better performance, according to equation (1).
4. Set the global best $\widehat{\mathbf{y}}^{(t)}$ to the position of the particle with the best performance within the swarm, according to equation (2).
5. Change the velocity vector for each particle, according to equation (3).
6. Move each particle to its new position, according to equation (4).
7. Let $t := t + 1$.
8. Go to step 2, and repeat until convergence.

The standard PSO algorithm described above does not lend itself well to optimising constrained functions. In particular, PSO can very easily be extended to optimise functions with linear equality constraints.

3 Linear Particle Swarm Optimisation

A new PSO algorithm, the Linear PSO, is developed specifically with linear constraints in mind. Traditionally, the velocity and position update equations, shown in equations (3) and (4), are specified separately for each dimension of a particle. If the random numbers $r_1^{(t)}$ and $r_2^{(t)}$ are rather kept constant for all vector dimensions, the velocity updates are calculated as a linear combination of position and velocity vectors.

$$\begin{aligned} \mathbf{v}_i^{(t+1)} &= w\mathbf{v}_i^{(t)} + c_1 r_1^{(t)}[\mathbf{y}_i^{(t)} - \mathbf{x}_i^{(t)}] \\ &\quad + c_2 r_2^{(t)}[\widehat{\mathbf{y}}^{(t)} - \mathbf{x}_i^{(t)}] \end{aligned} \quad (5)$$

$$\mathbf{x}_i^{(t+1)} = \mathbf{v}_i^{(t+1)} + \mathbf{x}_i^{(t)} \quad (6)$$

The above approach has the advantage that the flight of particles is defined by standard linear operations on vectors. The guaranteed movement of particles through vector spaces becomes possible, and hence the PSO algorithm using update equations (5) and (6) is referred to as a *Linear Particle Swarm Optimiser* (LPSO).

The LPSO ideally lends itself to optimising functions constrained by a set of linear constraints

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix}$$

thus solving problems of the form

$$\begin{aligned} &\text{Minimise} \quad f(\mathbf{x}), \quad \mathbf{x} \in \mathbb{R}^n \\ &\text{Subject to} \quad A\mathbf{x} = \mathbf{b}, \quad A \in \mathbb{R}^{m \times n} \text{ and } \mathbf{b} \in \mathbb{R}^m \end{aligned} \quad (7)$$

The above problem is equivalent to minimising f in hyperplane C, the set of particular solutions of the linear system $A\mathbf{x} = \mathbf{b}$. That is, $C = \{\mathbf{x} \mid A\mathbf{x} = \mathbf{b}\}$ defines the set of feasible solutions. Three changes to Algorithm 1 make it possible for the swarm of particles to only fly through C:

1. $\mathcal{S}$ is initialised such that the position $\mathbf{x}_i^{(0)}$ of each of the s particles meets $A\mathbf{x}_i^{(0)} = \mathbf{b}$. All initial velocity vectors are initialised as $\mathbf{v}_i^{(0)} = \mathbf{0}$;
2. Velocity update equation (5) is used; and
3. Position update equation (6) is used.

By implementing the above changes, the new PSO is guaranteed to *always* meet the set of constraints [13], and no more constraint handling is necessary. The LPSO algorithm is, however, not yet ready for practical use. This is because the problem of premature convergence has to be overcome.

Overcoming premature convergence

The LPSO algorithm discussed above has one property that is very disadvantageous, and that is the possibility of premature convergence.

If $\mathbf{v}^{(0)}$ is initialised to $\mathbf{0}$ and the position of the global best particle does not change, searches will continue on lines connecting each particle with the global best. Thus the entire hyperplane C will not be searched, but only lines connecting each particle to the global best.

In another scenario, consider $\mathbf{x}_i = \mathbf{y}_i = \widehat{\mathbf{y}}$, where velocity updates will depend only on the value of $w\mathbf{v}_i^{(t)}$, as discussed in [15, 16]. If a particle's current position coincides with the global best position, the particle will only move away from this point if its previous velocity and w are non-zero. Premature convergence will occur when previous velocities are close to zero, and particles stop moving once they catch up with the global best particle.

To overcome this premature convergence, the Guaranteed Convergence Particle Swarm Optimiser (GCPSO) was developed [15, 16]. In this algorithm, the velocity update for the global best particle is changed to force it to search for a better solution in an area around the position of that particle.

A variation of GCPSO, which alters particles only with feasible directions (such that the search will remain in C), is presented. The new algorithm, referred to as *Converging LPSO* (CLPSO), ensures that the constraints from equation (7) are still met. Let τ be the index of the global best particle, then

$$\mathbf{y}_\tau = \widehat{\mathbf{y}} \tag{8}$$

Change the velocity update equation (5) for the global best particle τ, so that

$$\mathbf{v}_\tau^{(t+1)} = -\mathbf{x}_\tau^{(t)} + \widehat{\mathbf{y}}^{(t)} + \rho^{(t)}\boldsymbol{\upsilon}^{(t)} \tag{9}$$

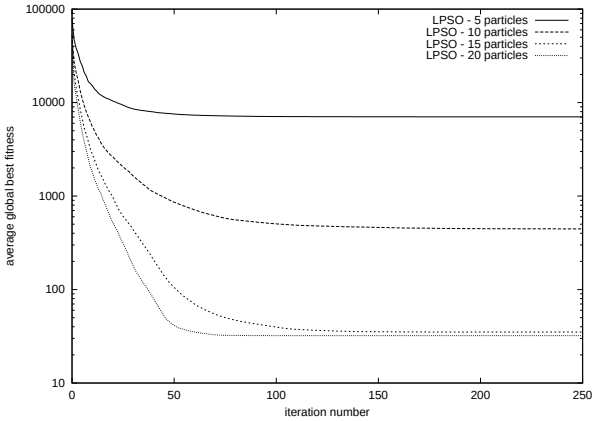
where $\rho^{(t)}$ is a scaling factor and $\boldsymbol{\upsilon}^{(t)} \sim \mathit{UNIF}(-1, 1)^n$ with the property that $A\boldsymbol{\upsilon}^{(t)} = \mathbf{0}$, or $\boldsymbol{\upsilon}^{(t)}$ lies in the null space of A. Since

$$\begin{aligned} \mathbf{x}_\tau^{(t+1)} &= \mathbf{v}_\tau^{(t+1)} + \mathbf{x}_\tau^{(t)} \\ &= \widehat{\mathbf{y}}^{(t)} + \rho^{(t)}\boldsymbol{\upsilon}^{(t)} \end{aligned}$$

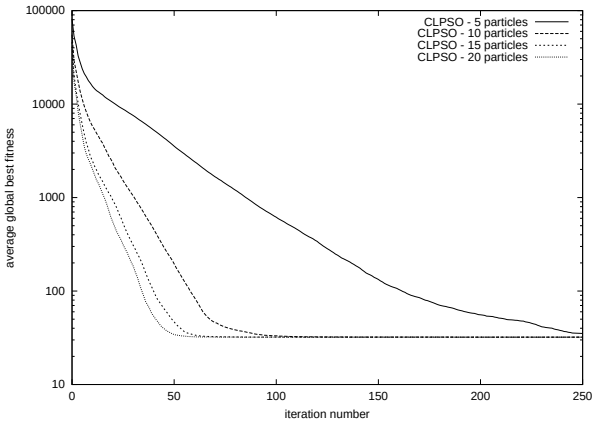
the new position of the global best particle will be its personal best $\widehat{\mathbf{y}}^{(t)}$, with a random vector $\rho^{(t)}\boldsymbol{\upsilon}^{(t)}$ from the null space of A added. It is *only* the global best particle that is moved with the above velocity update (9), all other particles in the swarm are still moved with the original equations (5) and (6).

It can be formally shown that CLPSO is guaranteed, in the limit, to always find at least a local minimum [13].

The difference between LPSO and CLPSO can be clearly seen from Figures 1(a) and 1(b). Both figures illustrate the average *gbest* values of the constrained parabola f_1



(a) LPSO



(b) CLPSO

Figure 1: Average *gbest* results of LPSO and CLPSO on the constrained parabola f_1 defined in equation (11). It is clearly shown how LPSO can converge prematurely, and how CLPSO always converges to at least a local minimum (which in this case is the actual minimum).

from equation (11). The averages are over 100 simulations, and show how LPSO can converge prematurely, and how CLPSO always converges to at least a local minimum (which in this case is the actual minimum).

4 Experimental results

In order to test the performance of LPSO and CLPSO to minimising problems constrained by a set of linear constraints $A\mathbf{x} = \mathbf{b}$, let

$$A = \begin{pmatrix} 0 & -3 & -1 & 0 & 0 & 2 & -6 & 0 & -4 & -2 \\ -1 & -3 & -1 & 0 & 0 & 0 & -5 & -1 & -7 & -2 \\ 0 & 0 & 1 & 0 & 0 & 1 & 3 & 0 & -2 & 2 \\ 2 & 6 & 2 & 2 & 0 & 0 & 4 & 6 & 16 & 4 \\ -1 & -6 & -1 & -2 & -2 & 3 & -6 & -5 & -13 & -4 \end{pmatrix}$$

$$\mathbf{b} = \begin{pmatrix} 3 & 0 & 9 & -16 & 30 \end{pmatrix}^T \quad (10)$$

Defining matrix A and vector $\mathbf{b}$ in the above way gives a set of constraints for testing ten-dimensional functions.

In all experiments the inertia weight w was set to 0.7, while the values of c_1 and c_2 were set to 1.4. The choice is due to [2], where it was shown that parameter settings close to these ($w = 0.7298$ and $c_1 = c_1 = 1.49618$) gave acceptable results. The value of $\rho^{(t)}$ was kept constant at 1. In each case, $\mathcal{S}$ was randomly initialised such that $A\mathbf{x}^{(0)} = \mathbf{b}$ holds for each particle. For test cases f_1, f_2, and f_3 (defined below), $\mathcal{S}$ was initialised in the interval [-100, 100]. For f_4, $\mathcal{S}$ was initialised in [2.56, 5.12], and $\mathcal{S}$ was initialised in [300, 600] for f_5.

Five constrained functions were chosen to be minimised, and include both convex functions and functions with many local minima. Function f_1 is a parabola (11), f_2 is a quadratic function (12), f_3 is a Rosenbrock function (13), f_4 is a Rastrigin function (14), and f_5 is a Griewank function (15). The test problems, with $\mathbf{x} \in \mathbb{R}^{10}$, are to minimise

$$f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \quad (11)$$

$$f_2(\mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{n} e^{-(x_i - x_j)^2} x_i x_j + \sum_{i=1}^{n} x_i \quad (12)$$

$$f_3(\mathbf{x}) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2) \quad (13)$$

$$f_4(\mathbf{x}) = \sum_{i=1}^{n} \left(x_i^2 - 10\cos(2\pi x_i) + 10\right) \quad (14)$$

$$f_5(\mathbf{x}) = \frac{1}{4000} \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (15)$$

Table 1: Results of 100 *Genocop II* simulations, and 100 LPSO and CLPSO simulations, on the constrained parabola f_1 defined in equation (11), after 250 generations or iterations. ('chromosomes' is abbreviated as chrms.)

Genocop II	10 chrms.	20 chrms.
Average	304.884	54.846
Maximum	1.168×10^3	107.584
Minimum	37.612	32.544
Standard Deviation	387.746	16.939
LPSO	10 particles	20 particles
Average	445.316	32.137
Maximum	4.505×10^3	32.137
Minimum	32.137	32.137
Standard Deviation	803.006	7.176×10^{-12}
CLPSO	10 particles	20 particles
Average	32.139	32.137
Maximum	32.183	32.137
Minimum	32.137	32.137
Standard Deviation	6.689×10^{-3}	3.016×10^{-6}

subject to $A\mathbf{x} = \mathbf{b}$, where A and $\mathbf{b}$ are defined in equation (10).

The correctness of the results are tested against those found by *Genocop II*, a genetic algorithm for optimising constrained problems [10]. For purposes of comparison with LPSO and CLPSO, *Genocop II* was evolved with the same number of chromosomes (particles) and generations (iterations) as LPSO and CLPSO.

One hundred simulations were done with *Genocop II* for each test function, where the genetic algorithm was evolved with a population size of 10 and 20 chromosomes. Experimental results on LPSO and CLPSO are also taken form a total of 100 simulations on swarm sizes of 10 and 20 particles.

The average best fitness values of *Genocop II* at the final generation, over all 100 simulations, are shown respectively for f_1, f_2, f_3, f_4, and f_5 in Tables 1, 2, 3, 4, and 5. The average *gbest* at the final iteration is similarly computed for LPSO and CLPSO, and also shown in the Tables.

The final maximum and minimum values are also shown for each test function, with the maximum being the largest of the 100 best fitness (or *gbest*) values at the final iteration, and the minimum being the smallest fitness (or *gbest*) value of the 100 simulations at the final iteration.

The standard deviation of *Genocop II*'s best fitness (or LPSO and CLPSO's *gbest*) values at the final iteration, computed over all 100 simulations, are also shown in the Tables.

The number of the final generation differs from problem to problem, and is indicated in the Table corresponding to each test function.

CLPSO's convergence to at least a local minimum is

Table 2: Results of 100 *Genocop II* simulations, and 100 LPSO and CLPSO simulations, on the constrained quadratic function f_2 defined in equation (12), after 1000 generations or iterations. ('chromosomes' is abbreviated as chrms.)

Genocop II	10 chrms.	20 chrms.
Average	49.945	39.500
Maximum	82.221	56.613
Minimum	35.393	35.410
Standard Deviation	10.996	6.785
LPSO	10 particles	20 particles
Average	758.525	59.762
Maximum	1.123×10^4	246.905
Minimum	35.400	35.377
Standard Deviation	1.496×10^3	39.831
CLPSO	10 particles	20 particles
Average	68.570	39.832
Maximum	196.067	71.380
Minimum	35.377	35.377
Standard Deviation	53.865	10.887

Table 3: Results of 100 *Genocop II* simulations, and 100 LPSO and CLPSO simulations, on the constrained Rosenbrock function f_3 defined in equation (13), after 2000 generations or iterations. ('chromosomes' is abbreviated as chrms.)

Genocop II	10 chrms.	20 chrms.
Average	21630.020	21485.714
Maximum	22030.988	21486.646
Minimum	21490.840	21485.363
Standard Deviation	154.443	0.400
LPSO	10 particles	20 particles
Average	4.444×10^6	1.260×10^5
Maximum	2.177×10^8	1.045×10^7
Minimum	21554.158	21485.925
Standard Deviation	2.278×10^7	1.043×10^6
CLPSO	10 particles	20 particles
Average	7.446×10^5	21485.305
Maximum	7.112×10^7	21485.305
Minimum	21485.305	21485.305
Standard Deviation	7.120×10^6	9.401×10^{-8}

Table 4: Results of 100 *Genocop II* simulations, and 100 LPSO and CLPSO simulations, on the constrained Rastrigin function f_4 defined in equation (14), after 1000 generations or iterations. ('chromosomes' is abbreviated as chrms.)

Genocop II	10 chrms.	20 chrms.
Average	52.379	43.059
Maximum	67.564	59.959
Minimum	37.116	37.011
Standard Deviation	7.498	6.142
LPSO	10 particles	20 particles
Average	76.487	75.011
Maximum	232.979	184.226
Minimum	36.975	38.965
Standard Deviation	30.699	27.719
CLPSO	10 particles	20 particles
Average	69.039	76.896
Maximum	154.379	151.394
Minimum	36.975	36.975
Standard Deviation	21.591	27.304

Table 5: Results of 100 *Genocop II* simulations, and 100 LPSO and CLPSO simulations, on the constrained Griewank function f_5 defined in equation (15), after 1000 generations or iterations. ('chromosomes' is abbreviated as chrms.)

Genocop II	10 chrms.	20 chrms.
Average	0.702	0.584
Maximum	0.971	0.843
Minimum	0.417	0.201
Standard Deviation	0.187	0.131
LPSO	10 particles	20 particles
Average	2.997	1.695
Maximum	15.805	14.401
Minimum	0.387	0.338
Standard Deviation	2.945	1.921
CLPSO	10 particles	20 particles
Average	3.049	1.900
Maximum	16.427	17.259
Minimum	0.236	0.236
Standard Deviation	3.101	2.379

clearly illustrated in Table 1 and Figure 1(b).

If the averages from all five tables are compared to the minimum values from all simulations, then the following can be deduced:

For convex functions, or functions without too many local minima, CLPSO performs considerably better than LPSO. This better performance can be ascribed to LPSO's possibility of premature convergence, without reaching a local or global minimum (see Tables 1, 2, and 3). The average *gbest* is also very similar to the best function evaluation found. The small standard deviation of *gbest* on the final iteration (especially in Tables 1 and 3) indicates that the swarms have converged to the same point in search space, or the same minimum.

Test functions f_4 and f_5 have many local minima. In this case, CLPSO's guarantee to at least converge to a local minimum implies that at least one of the numerous local minima will be found (which may differ substantially from the global minimum). The results in Tables 4 and 5 thus show similarity between LPSO and CLPSO. The average *gbest* also differs considerably from the best function evaluation found. The large standard deviation of *gbest* on the final iteration indicates that the swarms have found different local minima.

Genocop II, with a larger amount of mutation, performs better than CLPSO on functions with many local minima (see Tables 4 and 5), while CLPSO gives better performance on convex functions, where less mutation is needed (see Table 1).

To complete the experimental results, the best minimums found by CLPSO are shown, together with the domain values that attain these minimums.

Best solutions – CLPSO

Let $\mathbf{x}^\star$ indicate the vector found that gives the smallest evaluation for each test function. The very best performance of CLPSO over all 100 simulations, are shown here.

The minimum *gbest* of the constrained f_1, found by CLPSO, was $f_1(\mathbf{x}^\star) = 32.137$ with

$$\mathbf{x}^\star = (0.566, -0.485, 1.738, -1.181, -3.402, \ldots \\ \ldots 3.357, 0.900, -1.795, -0.528, 0.074)^T$$

The minimum *gbest* of the constrained f_2, found by CLPSO, was $f_2(\mathbf{x}^\star) = 35.377$ with

$$\mathbf{x}^\star = (0.076, -0.281, 0.445, -0.373, -3.956, \ldots \\ \ldots 3.762, 1.120, -1.865, -0.538, 0.178)^T$$

The minimum *gbest* of the constrained f_3, found by CLPSO, was $f_3(\mathbf{x}^\star) = 21485.305$ with

$$\mathbf{x}^\star = (0.840, -1.514, 2.359, -0.670, -3.352, \ldots \\ \ldots 2.991, 1.053, -1.949, -0.274, -0.028)^T$$

The minimum *gbest* of the constrained f_4, found by CLPSO, was $f_4(\mathbf{x}^\star) = 36.975$ with

$$\mathbf{x}^\star = (1.993, -0.002, 1.004, -1.996, -3.997, \ldots \\ \ldots 3.002, 0.999, -1.004, -0.998, 3.732)^T$$

The minimum *gbest* of the constrained f_5, found by CLPSO, was $f_5(\mathbf{x}^\star) = 0.236$ with

$$\mathbf{x}^\star = (0.076, -4.583, 10.670, -6.628, 0.225, \ldots \\ \ldots 0.919, 0.139, -0.818, 0.681, -0.821)^T$$

The experimental results presented here underline the simplicity with which PSO extends to the new linear PSO algorithms. Thus far, CLPSO has also been tested as a successful optimiser in Support Vector Machine training [12]. It also compares favourably to *Genocop II*, when linearly constrained functions are optimised.

5 Conclusion

A new PSO algorithm, the Linear PSO (LPSO) was developed to optimise functions constrained by linear constraints of the form $A\mathbf{x} = \mathbf{b}$. The LPSO changes the velocity and position updates of the traditional PSO algorithm, such that the guaranteed movement of particles through vector spaces becomes possible. The LPSO is a constraint-preserving method, and adds almost no extra cost of implementation to the traditional PSO. The probability of premature convergence of the LPSO is remedied by the extension to the Converging LPSO (CLPSO), which is guaranteed to always find at least a local minimum. The success and simplicity of LPSO and CLPSO in optimising linearly constrained functions was experimentally verified.

Acknowledgment

The financial assistance of the National Research Foundation towards this research is hereby acknowledged. Opinions expressed in this paper and conclusions arrived at, are those of the authors and not necessarily to be attributed to the National Research Foundation.

Bibliography

[1] L. Davis. *Genetic Algorithms and Simulated Annealing*. Pitman, London, 1987.

[2] R.C. Eberhart and Y. Shi. "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation*, pages 84-88. 2000.

[3] D.E. Goldberg. *Genetic Algorithms in Search Optimization and Machine Learning*. Addison-Wesley, 1989.

[4] S.B. Hamida and M. Schoenauer. “ASHEA: New results using adaptive segregational handling,” in *IEEE World Congress on Computational Intelligence, Proceedings of the Congress on Evolutionary Computing*. Honolulu, Hawaii, 2002.

[5] J. Kennedy and R.C. Eberhart. “Particle swarm optimization,” in *Proceedings of the IEEE International Conference on Neural Networks, IV*, pages 1942-1948. 1995.

[6] J. Kennedy. “Small worlds and mega minds: effects of neighborhood topology on particle swarm performance,” in *Proceedings of the Congress of Evolutionary Computation, Washington DC, USA*, pages 1931-1938. 1999.

[7] J. Kennedy, R.C. Eberhart, and Y. Shi. *Swarm Intelligence*. Morgan Kaufmann Publishers, 2001.

[8] J. Kennedy and R. Mendes. “Population structure and particle swarm performance,” in *IEEE World Congress on Computational Intelligence, Proceedings of the Congress on Evolutionary Computing*. Honolulu, Hawaii, 2002.

[9] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Verlag, 1996.

[10] Z. Michalewicz and C.Z. Janikow. “GENOCOP: a genetic algorithm for numerical optimization problems with linear constraints,” in *Communications of the ACM*, volume 39, article no. 175. 1996.

[11] Z. Michalewicz and M. Schoenauer. “Evolutionary algorithms for constrained parameter optimization Problems,” in *Evolutionary Computation*, volume 4, pages 1-32. 1996.

[12] U. Paquet and A.P. Engelbrecht. “Training support vector machines with particle swarms,” in *Proceedings of the International Joint Conference on Neural Networks*. Portland, Oregon, 2003.

[13] U. Paquet and A.P. Engelbrecht. “Particle swarms for equality-constrained optimization,” submitted to *IEEE Transactions on Evolutionary Computation*.

[14] Y. Shi and R.C. Eberhart. “A modified particle swarm optimizer,” in *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 69-73. Piscataway, NJ, 1998.

[15] F. van den Bergh. *An analysis of particle swarm optimizers*. PhD Thesis, Department of Computer Science, University of Pretoria, 2002.

[16] F. van den Bergh and A.P. Engelbrecht. “A locally convergent particle swarm optimiser,” accepted for *IEEE conference on Systems, Man, and Cybernetics*. Tunisia, 2002.

[17] D. Whitley, V.S. Gordon, and K. Mathias. “Lamarcian evolution, the baldwin effect and function optimization,” in Y. Davidor, H-P Schwefel, and R. Männer, editors, *Proceedings of the Third Conference on Parallel Problem Solving from Nature*. Springer, 1996.