

Chapter 5

Experimental results

The purpose of the following chapter, presenting experimental results, is twofold: The convergence of Linear PSO (LPSO) and Converging LPSO (CLPSO) is tested, and the CLPSO is implemented as the constrained optimisation algorithm that is used in training a Support Vector Machine (SVM).

Experimental results are shown to illustrate the differences between the LPSO and the CLPSO in linearly minimising constrained functions. The minima found by these two PSOs are compared for correctness against the minima found by a genetic algorithm implementation, called *Genocop II*.

As a conclusion, the CLPSO is used in the SVM training algorithm, defined in Section 2.4. The algorithm is empirically compared against two standard SVM training methods, namely decomposition and sequential minimal optimisation.

5.1 Linear Particle Swarm Optimiser

5.1.1 Experimental results

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

$$A = \begin{pmatrix} 0 & -3 & -1 & 0 & 0 & 2 & -6 & 0 & -4 & -2 \\ -1 & -3 & -1 & 0 & 0 & 0 & -5 & -1 & -7 & -2 \\ 0 & 0 & 1 & 0 & 0 & 1 & 3 & 0 & -2 & 2 \\ 2 & 6 & 2 & 2 & 0 & 0 & 4 & 6 & 16 & 4 \\ -1 & -6 & -1 & -2 & -2 & 3 & -6 & -5 & -13 & -4 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} 3 \\ 0 \\ 9 \\ -16 \\ 30 \end{pmatrix} \quad (5.1)$$

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 [18], where it is shown that parameter settings close to these ($w = 0.7298$ and $c_1 = c_2 = 1.49618$) give acceptable results. The value of $\rho^{(t)}$ was kept constant at 1.

The correctness of the results are tested against those found by *Genocop II*, a genetic algorithm for optimising constrained problems [32]. Experiments on *Genocop II* are done in a twofold manner:

1. A good minimum is needed against which comparisons can be made. In each case a good minimum for each constrained function was found by evolving the genetic algorithm with a population size of 100, for a total of 4000 generations.
2. For purposes of comparison with LPSO and CLPSO, *Genocop II* was also evolved with the same number of chromosomes (particles) and generations (iterations) as LPSO and CLPSO.

In the following experimental results, the 'good minimum' found by *Genocop II* (with a population size of 100 and after 4000 generations) is indicated first. After the good minimum is shown, the simulations used for comparison with LPSO and CLPSO are discussed.

Test 1

The first function tested, f_1 , is a second order polynomial (parabolic) function. For purposes of testing the free dimensions were randomly initialised in the interval $[-100, 100]$. The problem is defined as

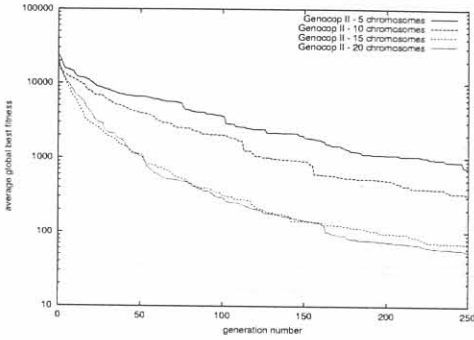
$$\begin{aligned} &\text{Minimise } f_1(\mathbf{x}) = \sum_i x_i^2, \mathbf{x} \in \mathbb{R}^{10} \\ &\text{Subject to } \quad A\mathbf{x} = \mathbf{b} \end{aligned} \tag{5.2}$$

where A and \mathbf{b} are defined in equation (5.1).

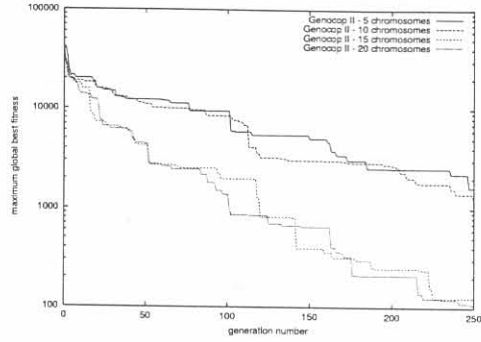
Genocop II The best solution found by *Genocop II*, with a population size of 100 and 4000 generations, was $f_1(\mathbf{x}^*) = 32.137$ with

$$\mathbf{x}^* = (0.567, -0.487, 1.736, -1.181, -3.404, 3.357, 0.9, -1.795, -0.528, 0.075)^T$$

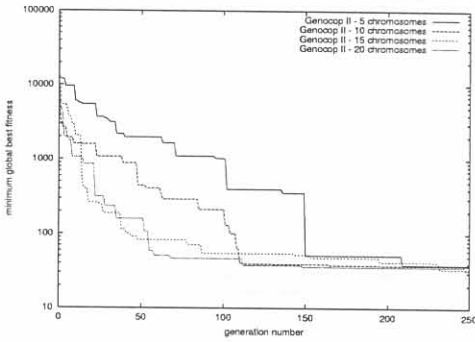
Genocop II was evolved for a total of 250 generations, for population sizes of 5, 10, 15, and 20 chromosomes. The average convergence over 100 simulations is shown in Figure 5.1(a). The average is determined over the best fitness values at a specific generation, over all simulations. The maximum and minimum values over all simulations are computed in a similar fashion, and are shown in Figures 5.1(b) and 5.1(c) respectively. The decreasing



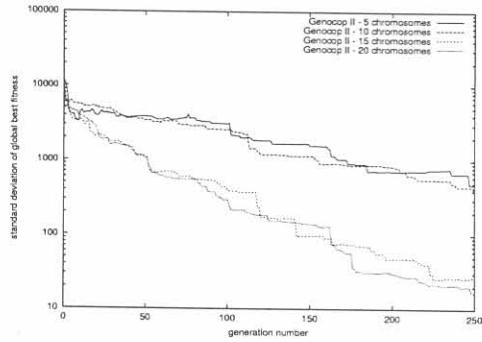
(a) Average



(b) Maximum



(c) Minimum



(d) Standard Deviation

FIGURE 5.1: Results of 100 *Genocop II* simulations on the constrained parabola f_1 defined in equation (5.2).

TABLE 5.1: Results of 100 *Genocop II* simulations on the constrained parabola f_1 defined in equation (5.2), after 250 generations. ('chromosomes' is abbreviated as chrms.)

<i>Genocop II</i>	5 chrms.	10 chrms.	15 chrms.	20 chrms.
Average	739.438	304.884	69.154	54.846
Maximum	1.626×10^3	1.168×10^3	124.820	107.584
Minimum	38.322	37.612	33.837	32.544
Standard Deviation	840.279	387.746	26.749	16.939

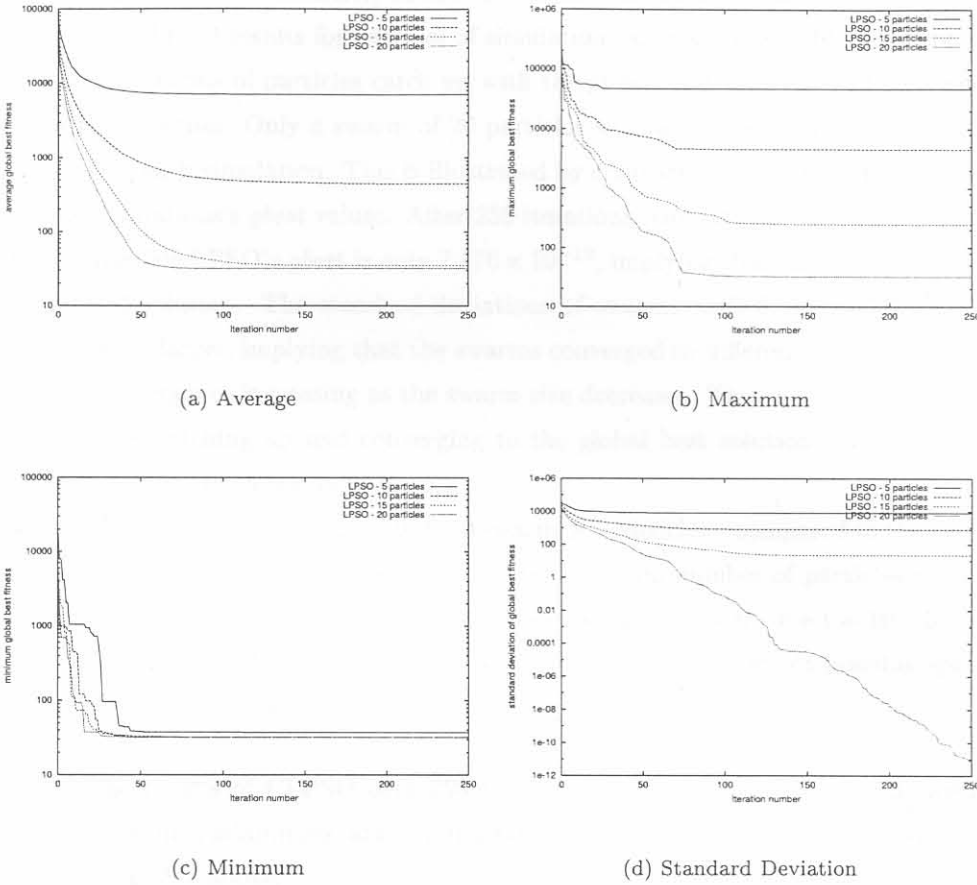


FIGURE 5.2: Results of 100 simulations of LPSO on the constrained parabola f_1 defined in equation (5.2).

maximum (worst) performances give a clear indication that the genetic algorithm converges for all simulations. The standard deviation of the best fitness values over 100 simulations is shown in Figure 5.1(d). These results are summarised in Table 5.1, and are compared to the PSO under the CLPSO results.

LPSO Figure 5.2 shows the convergence of LPSO over 250 iterations, or time steps, of the LPSO algorithm. The results are taken from a total of 100 simulations on swarm sizes of 5, 10, 15, and 20. The average at a specific iteration is computed over the 100 *gbest* values at that specific iteration number, and the averages over all iterations are illustrated in Figure 5.2(a). The maximums and minimums are computed in a similar way, with the maximum being the largest of the 100 *gbest* values at a specific iteration, and the minimum being the smallest of the 100 *gbest* values at a specific iteration. This is shown in Figures 5.2(b) and 5.2(c). The standard deviation of all the LPSO's *gbest* values at a certain time

(Figure 5.2(d)), shows the similarity in the convergence of the 100 swarms.

The average LPSO results for each set of simulations are completely different, and illustrates how the swarms of particles catch up with the global best particle, and converges to a sub-optimal solution. Only a swarm of 20 particles were able to converge to the optimal solution during each simulation. This is illustrated by comparing the standard deviations of each set of simulations's *gbest* values. After 250 iterations (time steps), the standard deviation the 20-particle LPSO's *gbest* is only 7.176×10^{-12} , implying that all swarms converged to the optimal solution. The standard deviations of swarms with 5, 10, and 15 particles are substantially larger, implying that the swarms converged to different solutions, with the variance in convergence increasing as the swarm size decreases. This is the expected result, due to particles catching up and converging to the global best solution [55]. The results after 250 iterations are shown in Table 5.2.

The large average *gbest* of 7.034×10^3 for a swarm of 5 particles – compared to the averages of 10, 15, and 20 particles – is also expected. The minimum number of particles needed to ensure that the swarm spans the entire search space, is $\inf |S^{(0)}| = n - r + 1 = 10 - 5 + 1 = 6$ (refer to equation (4.27)). Consequently, a swarm with 5 particles cannot possibly span the entire search space, which explains the large average *gbest*.

CLPSO The results of CLPSO over 250 time steps are shown in Figure 5.3, with the averages, maximums, minimums, and standard deviations computed in the same way as was done with the LPSO above.

The CLPSO simulations (for 5, 10, 15, and 20 particles) all converged on average to the minimum, or a value close to it. The minimum solution found was

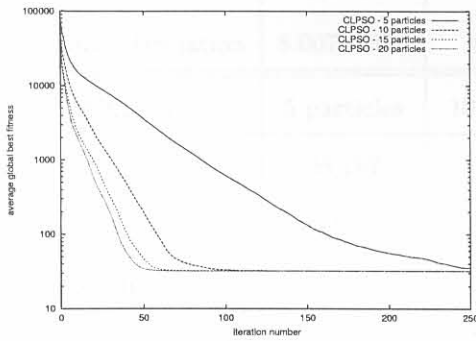
$$\mathbf{x}^* = (0.566, -0.485, 1.738, -1.181, -3.402, 3.357, 0.9, -1.795, -0.528, 0.074)^T$$

with

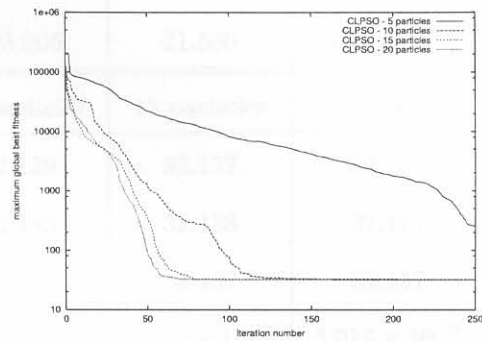
$$f_1(\mathbf{x}^*) = 32.137$$

The rate of convergence is higher for larger swarms. Figure 5.3(a) shows how the speed of convergence increases as the swarm size grows from 5 to 10, 15, and 20 particles. The standard deviations in Table 5.2 show that there is a very small variance in the *gbest* found by each swarm in the different sets of simulations, indicating that all swarms were close to or at the minimum solution after 250 time steps.

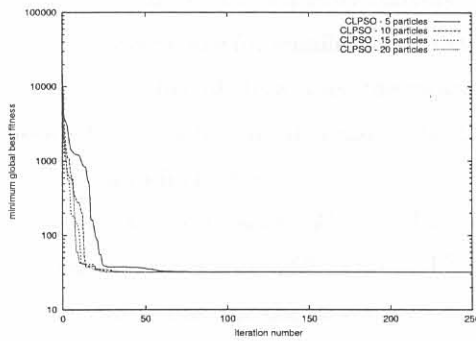
Since the initial condition (refer to equation (4.27)) on a swarm is dropped for the CLPSO, a swarm of 5 particles also searched the entire search space and found the minimum. This can be seen by comparing the average and minimum of a 5-particle swarm in Table 5.2. The difference between LPSO and CLPSO can be clearly seen when Figures 5.2(a) and



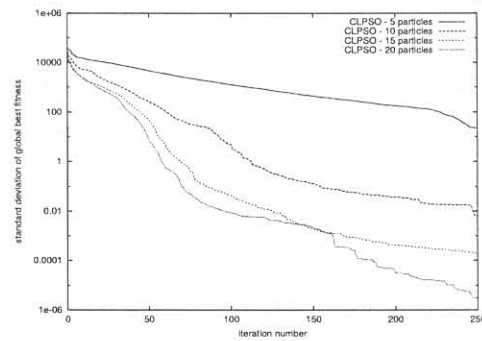
(a) Average



(b) Maximum



(c) Minimum



(d) Standard Deviation

FIGURE 5.3: Results of 100 simulations of CLPSO on the constrained parabola f_1 defined in equation (5.2).

TABLE 5.2: Results of 100 LPSO and CLPSO simulations on the constrained parabola f_1 defined in equation (5.2), after 250 iterations.

LPSO	5 particles	10 particles	15 particles	20 particles
Average	7.034×10^3	445.316	35.071	32.137
Maximum	4.630×10^4	4.505×10^3	244.077	32.137
Minimum	37.420	32.137	32.137	32.137
Standard Deviation	8.007×10^3	803.006	21.500	7.176×10^{-12}
CLPSO	5 particles	10 particles	15 particles	20 particles
Average	35.197	32.139	32.137	32.137
Maximum	252.826	32.183	32.138	32.137
Minimum	32.138	32.137	32.137	32.137
Standard Deviation	22.132	6.689×10^{-3}	1.832×10^{-4}	3.016×10^{-6}

5.3(a) are compared. The CLPSO converges on average to the minimum; the LPSO shows premature convergence for smaller swarm sizes, since when the global best does not improve over a large number of iterations, the swarm catches up with it. Note that the probability of finding better solutions increase with LPSO swarm size, and thus the probability of convergence also increases.

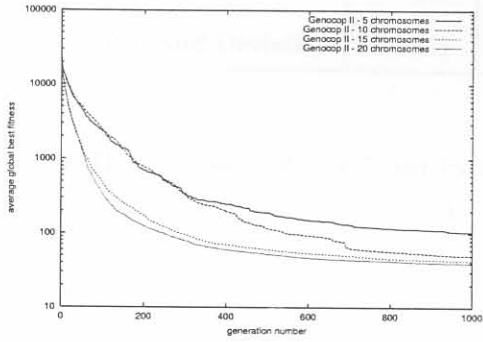
In comparison to *Genocop II*, the CLPSO has a substantially smaller standard deviation of *gbest* values at iteration 250. This is due to the fact that CLSPO has already converged, while *Genocop II* has, for the larger part of simulations, not yet converged to the minimum.

Test 2

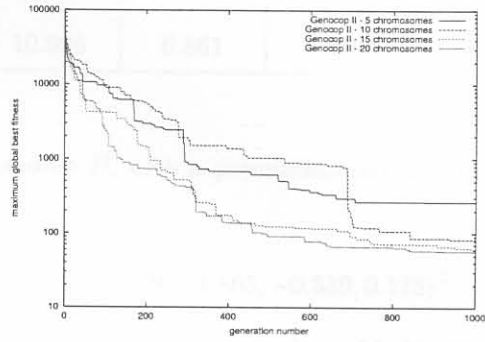
Function f_2 is a quadratic function similar to those commonly found in quadratic programming problems. This function was chosen because it is also similar to the dual Lagrangian optimised in SVM training. Again, the free dimensions were randomly initialised in the interval $[-100, 100]$. The problem is defined as

$$\begin{aligned}
 &\text{Minimise } f_2(\mathbf{x}) = \sum_i \sum_j e^{-(x_i - x_j)^2} x_i x_j + \sum_i x_i, \mathbf{x} \in \mathbb{R}^{10} \\
 &\text{Subject to } \quad \quad \quad \mathbf{Ax} = \mathbf{b}
 \end{aligned} \tag{5.3}$$

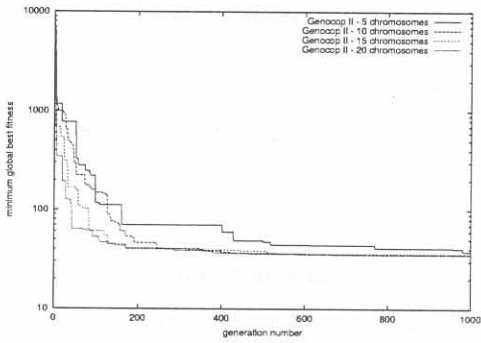
where A and \mathbf{b} are defined in equation (5.1).



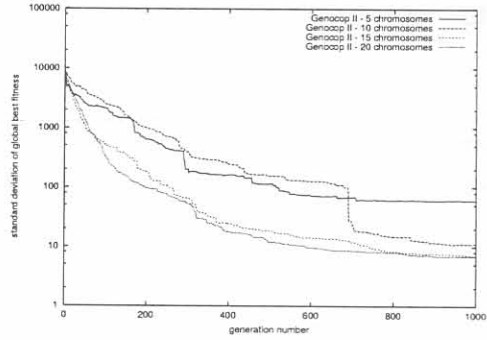
(a) Average



(b) Maximum



(c) Minimum



(d) Standard Deviation

FIGURE 5.4: Results of 100 *Genocop II* simulations on the constrained quadratic function f_2 defined in equation (5.3).

TABLE 5.3: Results of 100 *Genocop II* simulations on the constrained quadratic function f_2 defined in equation (5.3), after 1000 generations. ('chromosomes' is abbreviated as chrms.)

<i>Genocop II</i>	5 chrms.	10 chrms.	15 chrms.	20 chrms.
Average	104.192	49.945	42.393	39.500
Maximum	262.656	82.221	60.110	56.613
Minimum	37.939	35.393	35.772	35.410
Standard Deviation	59.873	10.996	6.861	6.785

Genocop II The best solution found by *Genocop II*, with a population size of 100 and 4000 generations, was $f_2(\mathbf{x}^*) = 35.377$ with

$$\mathbf{x}^* = (0.076, -0.28, 0.446, -0.373, -3.956, 3.762, 1.119, -1.865, -0.539, 0.178)^T$$

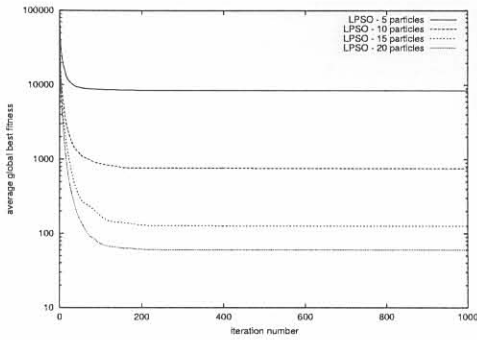
Genocop II was evolved for a total of 1000 generations, for population sizes of 5, 10, 15, and 20 chromosomes. The averages, maximums, minimums and standard deviations over 100 simulations are shown in Figure 5.4, and are computed in the same way as Test 1. Again, these results are summarised in Table 5.3, and are compared to the PSO under the CLPSO results.

LPSO The results of LPSO over 1000 time steps are shown in Figure 5.5, with the averages, maximums, minimums, and standard deviations computed in the same way as explained in Test 1 above.

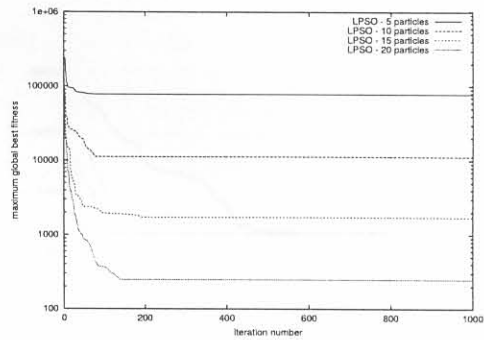
The averages, maximums and standard deviations illustrate the same behaviour as the results in Test 1 (optimising the constrained f_1 with LPSO). It is worthwhile to note that the minimum found by the LPSO, as seen in Figure 5.5(c) and Table 5.4, is the true minimum, except for the 5-particle case. This again illustrates that the LPSO's 5 particles do not span the entire search space, which is 6-dimensional.

CLPSO The results of CLPSO over 1000 time steps are shown in Figure 5.6, with the averages, maximums, minimums, and standard deviations computed in the same way as explained in Test 1 above.

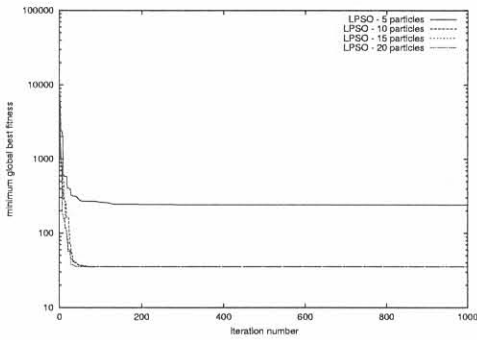
It is clear from Figure 5.6(a) that, after 1000 iterations, the CLSPO is still converging. After 2000 iterations (not shown in the figures), the average *gbest* values were 76.677 for 5 particles, 66.084 for 10 particles, 56.731 for 15 particles, and 39.537 for 20 particles. The averages after 2000 iterations are all smaller than the averages at 1000 generations, shown in Table 5.4.



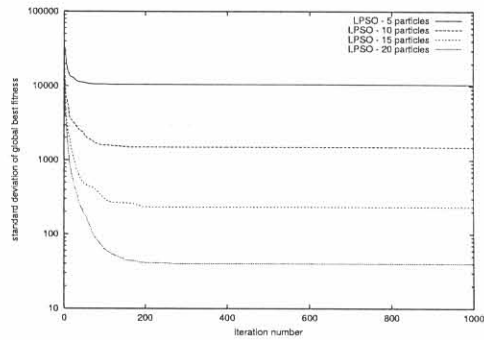
(a) Average



(b) Maximum

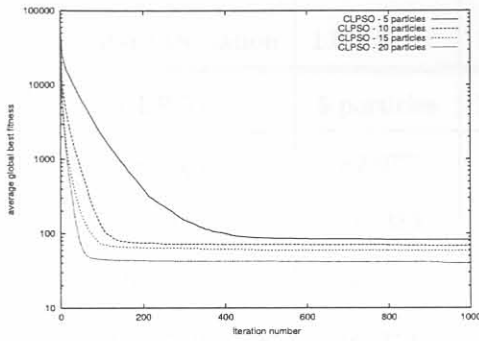


(c) Minimum

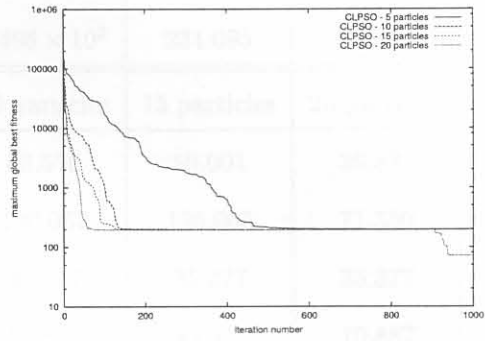


(d) Standard Deviation

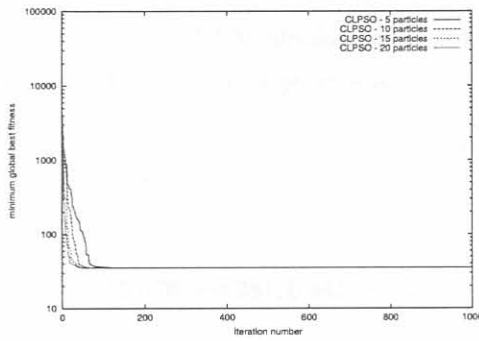
FIGURE 5.5: Results of 100 simulations of LPSO on the constrained quadratic function f_2 defined in equation (5.3).



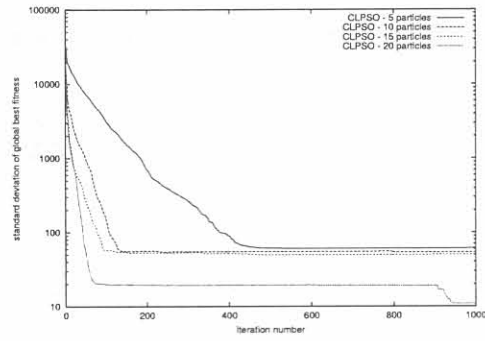
(a) Average



(b) Maximum



(c) Minimum



(d) Standard Deviation

FIGURE 5.6: Results of 100 simulations of CLPSO on the constrained quadratic function f_2 defined in equation (5.3).

TABLE 5.4: Results of 100 LPSO and CLPSO simulations on the constrained quadratic function f_2 defined in equation (5.3), after 1000 iterations.

LPSO	5 particles	10 particles	15 particles	20 particles
Average	8.463×10^3	758.525	125.727	59.762
Maximum	7.793×10^4	1.123×10^4	1.719×10^3	246.905
Minimum	240.101	35.400	35.377	35.377
Standard Deviation	1.051×10^4	1.496×10^3	231.095	39.831
CLPSO	5 particles	10 particles	15 particles	20 particles
Average	82.077	68.570	59.001	39.832
Maximum	197.389	196.067	196.065	71.380
Minimum	35.377	35.377	35.377	35.377
Standard Deviation	60.959	53.865	49.957	10.887

Table 5.4 illustrates the average, maximum, minimum, and standard deviation of the *gbest* convergence of 100 simulations of swarms with 5, 10, 15, and 20 particles, after 1000 time steps. The minimum *gbest* was

$$f_2(\mathbf{x}^*) = 35.377$$

at

$$\mathbf{x}^* = (0.076, -0.281, 0.445, -0.373, -3.956, 3.762, 1.12, -1.865, -0.538, 0.178)^T$$

If the average minimum values found in Figures 5.4(a) and 5.6(a) are compared, CLPSO shows a faster rate of convergence than *Genocop II*. The standard deviation after 1000 iterations or generations is smaller for *Genocop II* (compare Tables 5.3 and 5.4), indicating greater consistency in convergence between the different simulations.

Test 3

The third function tested, f_3 , is a Rosenbrock function in ten dimensions. The constrained f_3 differs from both f_1 and f_2 because it is not a convex function. The free dimensions were randomly initialised in the interval $[-100, 100]$. The problem is defined as

$$\begin{aligned} &\text{Minimise } f_3(\mathbf{x}) = \sum_{i=1}^{n-1} (100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2), \mathbf{x} \in \mathbb{R}^{10} \\ &\text{Subject to } \mathbf{Ax} = \mathbf{b} \end{aligned} \quad (5.4)$$

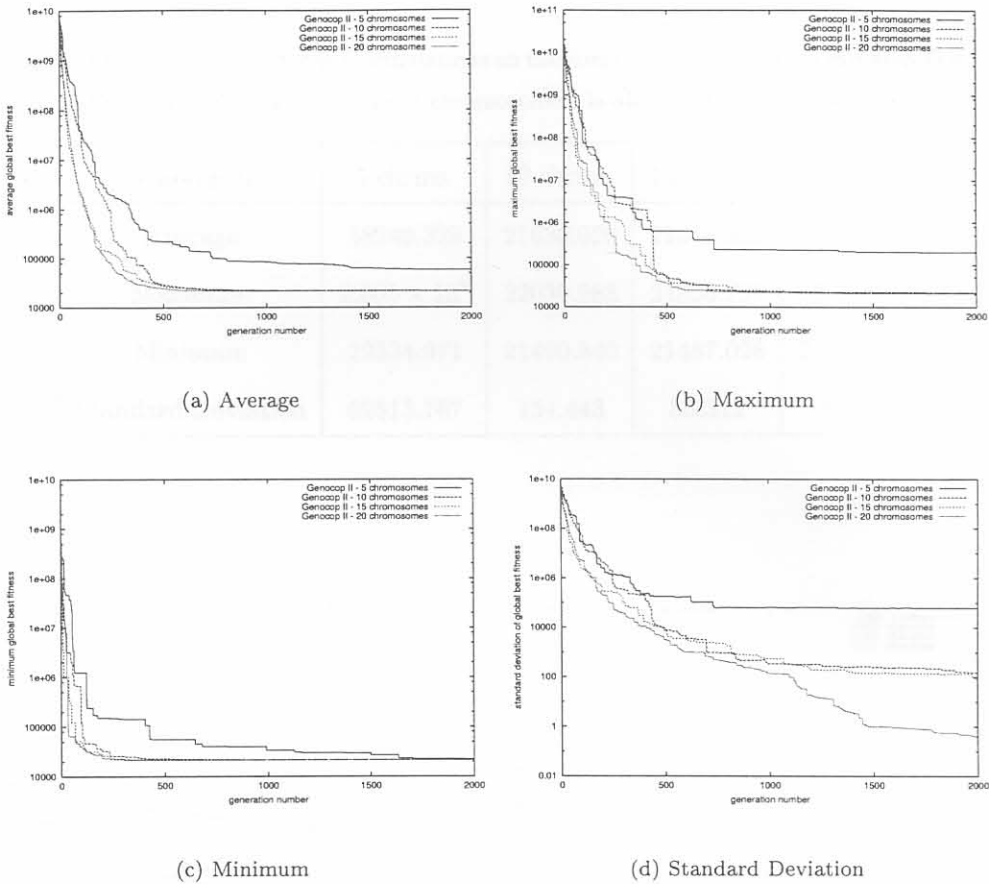


FIGURE 5.7: Results of 100 *Genocop II* simulations on the constrained Rosenbrock function f_3 defined in equation (5.4).

where A and b are defined in equation (5.1).

Genocop II The best solution found by *Genocop II*, with a population size of 100 and 4000 generations, was $f_3(\mathbf{x}^*) = 21485.361$ with

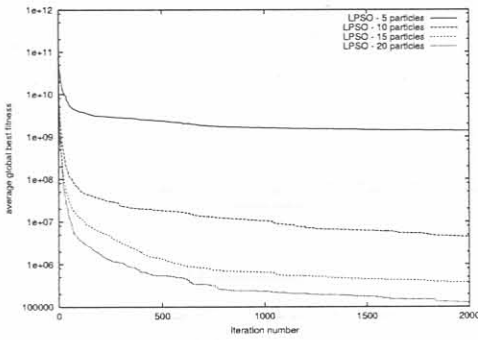
$$\mathbf{x}^* = (0.84, -1.516, 2.359, -0.669, -3.352, 2.991, 1.053, -1.949, -0.273, -0.028)^T$$

Genocop II was evolved for a total of 2000 generations, for population sizes of 5, 10, 15, and 20 chromosomes. The averages, maximums, minimums and standard deviations over 100 simulations are shown in Figure 5.7, and are computed in the same way as Test 1. Again, these results are summarised in Table 5.5, and are compared to the PSO under the CLPSO results.

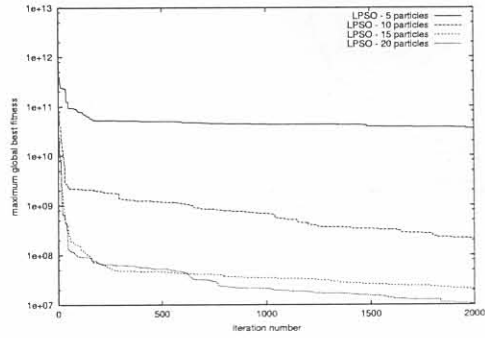
LPSO The results of LPSO over 2000 time steps are shown in Figure 5.8, with the averages, maximums, minimums, and standard deviations computed in the same way as explained in

TABLE 5.5: Results of 100 *Genocop II* simulations on the constrained Rosenbrock function f_3 defined in equation (5.4), after 2000 generations. ('chromosomes' is abbreviated as chrms.)

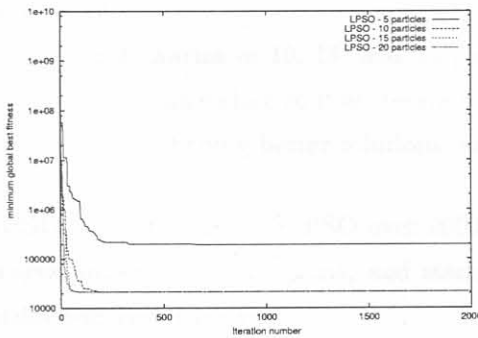
<i>Genocop II</i>	5 chrms.	10 chrms.	15 chrms.	20 chrms.
Average	58249.328	21630.020	21546.332	21485.714
Maximum	2.005×10^5	22030.988	21836.797	21486.646
Minimum	22334.971	21490.840	21487.098	21485.363
Standard Deviation	62513.767	154.443	85.311	0.400



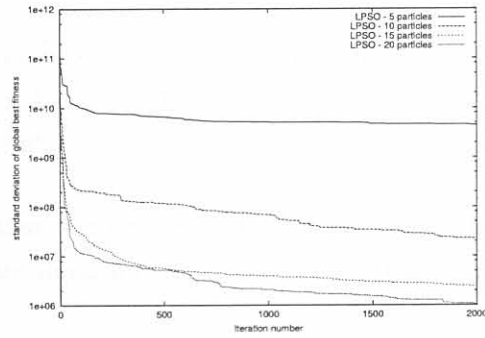
(a) Average



(b) Maximum



(c) Minimum



(d) Standard Deviation

FIGURE 5.8: Results of 100 simulations of LPSO on the constrained Rosenbrock function f_3 defined in equation (5.4).

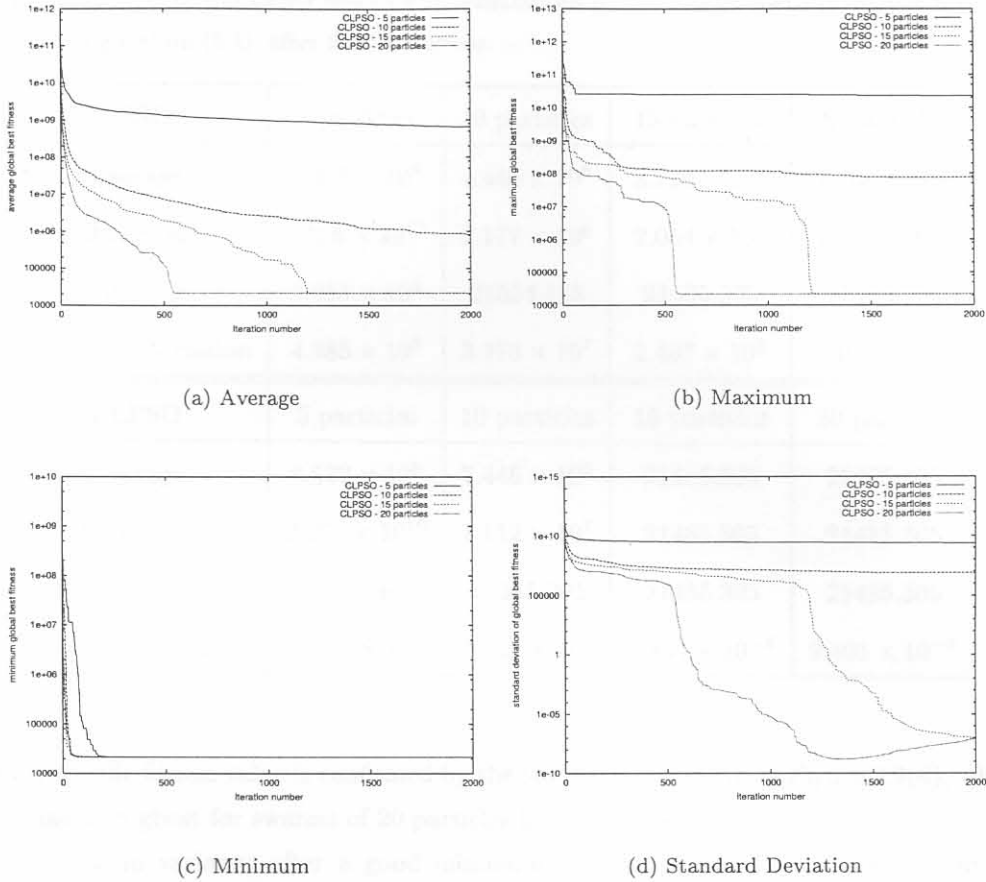


FIGURE 5.9: Results of 100 simulations of CLPSO on the constrained Rosenbrock function f_3 defined in equation (5.4).

Test 1 above.

The LPSO swarms of 10, 15, and 20 particles managed to find the minimum value of the function, or came close to it at iteration number 2000. After 2000 iterations, the LPSO swarms were still finding better solutions, as is illustrated in Figure 5.8(a).

CLPSO The results of CLPSO over 2000 time steps are shown in Figure 5.9, with the averages, maximums, minimums, and standard deviations computed in the same way as explained in Test 1 above.

The 20-particle CLSPO consistently converged to the minimum, as can be seen from Figure 5.9 and Table 5.6. Figure 5.9(a) also shows that the average fitness decreases dramatically to the minimum after the swarm has converged below a certain fitness level. It also shows that the swarms of 5 and 10 particles did not stagnate, but are still converging at the 2000th iteration. The sudden and complete convergence when the swarm decreases

TABLE 5.6: Results of 100 LPSO and CLPSO simulations on the constrained Rosenbrock function f_3 defined in equation (5.4), after 2000 iterations.

LPSO	5 particles	10 particles	15 particles	20 particles
Average	1.375×10^9	4.444×10^6	3.710×10^5	1.260×10^5
Maximum	3.556×10^{10}	2.177×10^8	2.054×10^7	1.045×10^7
Minimum	1.955×10^5	21554.158	21483.373	21485.925
Standard Deviation	4.485×10^9	2.278×10^7	2.407×10^6	1.043×10^6
CLPSO	5 particles	10 particles	15 particles	20 particles
Average	6.522×10^8	7.446×10^5	21485.305	21485.305
Maximum	2.233×10^{10}	7.112×10^7	21485.305	21485.305
Minimum	21485.306	21485.305	21485.305	21485.305
Standard Deviation	2.395×10^9	7.120×10^6	9.834×10^{-8}	9.401×10^{-8}

below a specific fitness value is confirmed by the standard deviations of Figure 5.9(d), where the variance in $gbest$ for swarms of 20 particles becomes close to zero.

The raise in variance after a good minimum was found (see Figure 5.9(d)), can be attributed to the random search performed by CLPSO. As minutely better minimums are found, the $gbest$ values will start to differ slightly, causing a rise in standard deviation in the order of 10^{-7} .

The CLPSO found

$$\mathbf{x}^* = (0.84, -1.514, 2.359, -0.67, -3.352, 2.991, 1.053, -1.949, -0.274, -0.028)^T$$

after 2000 time steps. The value of f_3 at \mathbf{x}^* was

$$f_3(\mathbf{x}^*) = 21485.305$$

The average best fitness of *Genocop II* is substantially better than that of both LPSO and CLPSO for small population or swarm sizes (see Figures 5.7(a) and 5.9(a)). This can be ascribed to a greater amount of mutation on the chromosomes (particles), and therefore a greater diversity in the solutions tested. The better convergence for small population or swarm sizes is supported by the difference in the standard deviations over the simulations, shown in Tables 5.5 and 5.6. For larger populations or greater swarm sizes, *Genocop II* and CLPSO have very similar performance.

5.1.2 LPSO and CLPSO Convergence characteristics

Some remarks, all confirming the theoretical properties needed for LPSO and CLPSO to successfully converge to a minimum, can be made from the above experimental results.

1. A swarm of 5 particles is smaller than the minimum swarm size, as derived in equation (4.27), of

$$\inf |S^{(0)}| = n - r + 1 = 10 - 5 + 1 = 6$$

Thus LPSO will not cover all search dimensions, and results can be expected to be suboptimal. Indeed, the average *gbest* (minimum) of the constrained f_1 in equation (5.2) was at 7.034×10^3 after 250 time steps, while CLPSO managed to find an average *gbest* of 35.197 with five particles. The comparison is shown in Table 5.2. With five particles and $t = 1000$, LPSO's average best for f_2 defined in (5.3) was 8.463×10^3 , while CLPSO's best f_2 was 82.007 (see Table 5.4).

2. As can be clearly seen in Figures 5.2, 5.5, and 5.8, the swarm catches up with the global best particle before reaching a minimum to cause premature convergence. This problem is overcome by CLPSO, as the empirical results in Figures 5.3, 5.6, and 5.9 illustrate.

5.2 Support Vector Machine Training

After showing the convergence and properties of the newly developed LPSO and CLPSO, the CLPSO algorithm will be implemented in training SVMs. This section illustrates the success and simplicity of the method, and also discusses some bottlenecks that have to be overcome to make the algorithm practically competitive.

5.2.1 Implementing the SVM training algorithm

Two issues remain to be resolved in implementing the SVM training algorithm described in Section 2.4. Both issues consist of finding feasible vectors: The first is to find an initial feasible solution α for the algorithm to start with. The second is, given a working set B , to initialise the swarm of particles that is going to optimise B , such that the swarm is feasible.

Finding an initial feasible solution α

To resolve the first issue, a feasible solution that satisfies the linear constraint $\alpha^T \mathbf{y} = 0$, with constraints $0 \leq \alpha_i \leq C$ also met, is needed at the start of the decomposition algorithm. The initial solution is constructed in the following way:

Let c be some real number between 0 and C , and γ some positive integer less than both the number of positive examples ($y_i = +1$) and negative examples ($y_i = -1$) in the training set. Randomly pick a total of γ positive examples, and γ negative examples, and initialise their corresponding α_i to c . By setting all other α_i to zero, the initial solution will be feasible.

The value 2γ gives the total number of initial support vectors, and since these initial support vectors are a randomly chosen guess, it is suggested that the value of γ be kept small.¹

Initialising a feasible swarm of particles

To resolve the second issue, consider the constrained optimisation problem solved by the CLPSO, repeated here for convenience:

$$\max_{\alpha_B} W(\alpha_B) = \alpha_B^T \mathbf{1} - \frac{1}{2} \alpha_B^T Q_{BB} \alpha_B - \alpha_B^T Q_{BN} \alpha_N \quad (5.5)$$

subject to

$$\begin{aligned} \alpha_B^T \mathbf{y}_B + \alpha_N^T \mathbf{y}_N &= 0 \\ \alpha_B &\geq 0 \\ C\mathbf{1} - \alpha_B &\geq 0 \end{aligned} \quad (5.6)$$

In optimising the q -dimensional subproblem, CLPSO requires that all particles be initialised such that $\alpha_B^T \mathbf{y}_B + \alpha_N^T \mathbf{y}_N = 0$ is met. This is done as follows:

1. Set each particle in the swarm to the q -dimensional vector α_B .
2. Add a random q -dimensional vector δ satisfying $\mathbf{y}_B^T \delta = 0$ to each particle, under the condition that the particle will still lie in the hypercube $[0, C]^q$.

The complexity of generating a q -dimensional random vector δ is $\mathcal{O}(q)$, since $q - 1$ components of δ can be chosen randomly, and the q^{th} component will δ_q will be $\delta_q = (\mathbf{y}_B)_q [- (\mathbf{y}_B)_1 \delta_1 - \dots - (\mathbf{y}_B)_{q-1} \delta_{q-1}]$. If the generation of a random number takes a constant amount of time, then initialising a swarm of s particles will be $\mathcal{O}(sq)$.

Initialising the swarm in this way ensures that the initial swarm lies in the set of feasible solutions $P = \{\mathbf{p} \mid A\mathbf{p} = -\alpha_N^T \mathbf{y}_N\}$, allowing the flight of the swarm to be defined by feasible directions.

¹In reality, *any* non-zero feasible vector can be used as an initial solution; choosing γ positive and negative examples only gives a simple way of constructing such a vector. Larger values of γ imply a larger set of initial support vectors, and the training algorithm simply spends extra time in removing the non-support vectors from this set.

5.2.2 Practical concerns and improvements

A number of practical issues need to be addressed to implement the algorithm numerically. One issue is on deciding when a solution is ‘optimal enough,’ and the Karush-Kuhn-Tucker conditions are adapted to be correct within an error threshold from the true conditions. The SVM training algorithm presented in Chapter 2 assumes infinite precision arithmetic. Since machine numbers allow only finite accuracy, the problem of error accumulation and round-off errors is addressed. A strategy is also given to optimise the dot product between two sparse vectors.

An approximation to the optimality conditions

The Karush-Kuhn-Tucker conditions (2.33) that define the stopping criteria for the training algorithm, specify that an $\alpha_i^{(t)}$ between zero and C must imply that $y_i(s_i^{(t)} + b^{(t)})$ should be exactly equal to one. In practice this is not always possible, and a small positive error ϵ on the KKT conditions will be tolerated to allow the algorithm to terminate. The value of ϵ close to 0.01 or 0.02 will typically give a very accurate optimisation [25]. The practical KKT conditions are therefore

$$\begin{aligned}
 \alpha_i^{(t)} = 0 &\Rightarrow y_i(s_i^{(t)} + b^{(t)}) > 1 - \epsilon \\
 0 < \alpha_i^{(t)} < C &\Rightarrow 1 - \epsilon < y_i(s_i^{(t)} + b^{(t)}) < 1 + \epsilon \\
 \alpha_i^{(t)} = C &\Rightarrow y_i(s_i^{(t)} + b^{(t)}) < 1 + \epsilon
 \end{aligned} \tag{5.7}$$

Error accumulation and round-off errors

The nature of the constrained LPSO algorithm allows for division and multiplication by very large and very small real numbers. This can give rise to numerical precision problems. One of the constraints on the SVM optimisation problem is that the sum of all $y_i\alpha_i$ must be equal to zero. It may be true that, due to rounding errors, this sum can shift from zero. To solve this problem, a check is done to determine

$$error = \sum_{i=1}^l y_i\alpha_i$$

To reset the sum to zero, one of the zero Lagrange multipliers α_i is set to the absolute value of *error*. If *error* is positive, an α_i corresponding to a negative example y_i is randomly chosen. If the opposite is true and *error* is negative, an α_i corresponding to a positive example y_i is randomly chosen. As optimisation continues, this adjusted Lagrange multiplier will be picked for reoptimisation, with the equality constraint holding.

The update is done when *error* rises above a certain threshold; in the experiments presented here, *error* was in the order of 10^{-6} . In practice this update rarely happens, but can occur.

Optimising the dot product between two sparse vectors

The time taken to compute the dot product between two sparse vectors can be greatly optimised if all multiplications with zero are simply ignored. The dot product between two n -dimensional vectors \mathbf{x}_i and \mathbf{x}_j is defined as

$$\mathbf{x}_i \cdot \mathbf{x}_j = x_{i1}x_{j1} + x_{i2}x_{j2} + \dots + x_{in}x_{jn}$$

Since a sparse vector contains many zero elements, many multiplications will be with zero and therefore unnecessary. The following algorithm is adapted from [42], and scans through both vectors to compute the dot product:

```
/* Array x1, with length n1, is an array that stores only
   xi's nonzero components. The original positions of these
   components in vector xi is stored in array id1. Arrays
   x2 and id2 with size n2 is used to store sparse vector xj.
*/
p1 = 0, p2 = 0, dot = 0
while (p1 < n1 && p2 < n2)
{
    a1 = id1[p1], a2 = id2[p2]
    if (a1 == a2)
    {
        dot += x1[p1]*x2[p2]
        p1++, p2++
    }
    else if (a1 > a2)
        p2++
    else
        p1++
}
```

5.2.3 Experimental results

The SVM training algorithm presented in Section 2.4 was tested on the MNIST dataset [34]. The influence of different working set sizes, as well as the scalability of the approach, is examined. Finally, the training results are compared to two other algorithms, a decomposition method and the method of sequential minimal optimisation.



FIGURE 5.10: A few examples from the MNIST dataset.

The MNIST dataset

The MNIST database is an optical character dataset, and consists of a training set of 60,000 handwritten digits [34]. This database is a subset of a larger set available from the National Institute of Standards Bureau (NIST). As shown in Figure 5.10, the examples are 28 by 28 pixel grey-level images. This is equivalent to each example being a 784-dimensional vector. Each pixel value corresponds to an integer in the range 0 (white) to 255 (black). It is a common database for benchmarking learning techniques and pattern recognition methods.

Training the SVM

For training a SVM on the MNIST dataset, the character ‘8’ was chosen to represent the set of positive examples, while the remaining digits defined the negative examples. Training was done with a polynomial kernel of degree five:

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + 1)^5 \quad (5.8)$$

Due to the size of the dot product between two images, raised to the fifth power, the pixel

TABLE 5.7: Influence of different working set sizes on the first 20,000 elements of the MNIST dataset

Working set size	Working set selections	Time	SVs
4	8,782	02:17:43	1,631
6	8,213	03:11:40	1,637
8	7,502	03:51:24	1,639
10	10,023	06:27:06	1,648
12	9,667	07:26:23	1,652

values were scaled to the range $[0, 0.1]$. This gives Lagrange multipliers α_i that are easier for the CLPSO to handle. (The kernel function of two unscaled black images would be $(784 \times 255^2 + 1)^5$, while the kernel function of the scaled versions gives a more practical $(784 \times 0.01 + 1)^5 \approx 835$).

For an optimal solution to be found in the following PSO experiments, the KKT conditions in equation (5.7) needed to be satisfied within an error threshold of $\epsilon = 0.02$. Optimisation of the working set terminated when the KKT conditions on the working set were met with an error of 0.001, or when the swarm has optimised for a hundred iterations.

The following parameters defined the experimental CLPSO: By letting $\gamma = 10$, a total of 20 initial support vectors were chosen to start the algorithm. The swarm size s used in each experiment was 10, while the inertia weight w was set to 0.7. The acceleration coefficients c_1 and c_2 were both set to 1.4 [55]. Since the objective function is constrained by a set of box constraints, the velocity vectors were not clamped. For each experiment the upper bound C was kept at 100.0 (a commonly used upper bound in SVM training).

The PSO training algorithm was written in Java, and does not make use of caching and shrinking methods to optimize its speed. The sparsity of input data is used to speed up the evaluation of kernel functions. All experiments were performed on a 1.00 GHz AMD Duron processor.

Experimental results show successful and accurate training on the MNIST database. The influence of different working set sizes on the CLPSO training algorithm, its scalability, as well as its relation to other SVM training algorithms, were examined.

TABLE 5.8: Scalability: training on the MNIST dataset

MNIST elements	PSO Working set selections	PSO time	PSO SVs	SMO time	SMO SVs	SVM ^{light} time	SVM ^{light} SVs
10,000	3,898	00:29:49	1,022	00:01:29	1,032	00:02:02	1,034
20,000	8,782	02:17:43	1,631	00:06:14	1,647	00:10:43	1,641
30,000	12,428	04:50:11	1,988	00:13:22	2,012	00:23:04	2,001
40,000	15,725	08:14:26	2,353	00:22:46	2,355	00:41:09	2,367
50,000	22,727	15:05:09	2,728	01:46:38	2,740	01:31:48	2,726
60,000	25,914	20:54:15	3,025	04:38:11	3,043	08:01:05	3,026

Influence of working set sizes

Experiments on different working set sizes were done on the first 20,000 elements of the MNIST database. Results are shown in Table 5.7, and indicate that a working set of size $q = 4$ gives the fastest convergence time and fewest support vectors. A working set of size 2 can be solved analytically, as is true in the case of Sequential Minimal Optimisation (SMO). The results in Table 5.7 are not necessarily an indication of the speed of the PSO on the working set, as selection of the working set also burdens the speed of the algorithm (the $\frac{q}{2}$ greatest and least values of $y_i \nabla W(\alpha)_i$ need to be selected from a list of thousands).

Scalability of the PSO approach

Scalability of the PSO algorithm was tested by training on the first 10,000, 20,000, etc. examples from the MNIST dataset, as shown in Table 5.8. In each case a working set of size 4 was used. The experimental results indicate that the PSO training algorithm shows quadratic scalability, and scales as $\sim l^{2.1}$ (with l being the training set size).

Comparison to other algorithms

In Table 5.8, the PSO approach is compared to SMO and a decomposition method, SVM^{light} [25]. WinSVM was developed by C. Longbin [30] from the SVM^{light} source code, and was used as an implementation of SMO. Unlike these methods, the current PSO algorithm does not make use of caching and shrinking to optimise its speed.

Results similar to Table 5.7 indicate that SVM^{light} gives the fastest rate of convergence with a working set size $q = 8$, which is used in Table 5.8's comparison.

Experimental results show SMO scaling as $\sim l^{2.8}$, and SVM^{light} scaling as $\sim l^{3.0}$. Both these algorithms are substantially faster than training a SVM with PSO on the MNIST dataset, but the PSO approach shows better scaling abilities ($\sim l^{2.1}$). Due to the fact that the PSO training algorithm starts with a very small set of possible support vectors, with all other α_i set to zero, the PSO method consistently finds a few support vectors less than the other approaches.

The main drawback from the current PSO approach is its slow performance times, but from this initial study many optimisations can be implemented on both decomposition and PSO methods.

5.3 Concluding

The success of the CLPSO in optimising linearly constrained functions was experimentally illustrated in this chapter. The necessity to change the LPSO to a locally converging algorithm was also shown.

It was shown that a PSO can be used to train a SVM. Some properties of LPSO make it particularly useful to solve the SVM constrained QP problem. The PSO algorithm is simple to implement, and does not require any background of numerical methods. Accurate and scalable training results were shown on the MNIST dataset, with the PSO algorithm finding fewer support vectors and better scalability than other approaches. Although the algorithm is simple, its speed poses a practical bottleneck, which can be improved from this initial study.