

Particle Swarm Optimisation

by

Hayley Cronin



Thesis

Submitted by Hayley Cronin

in partial fulfillment of the Requirements for the Degree of

Bachelor of Software Engineering with Honours (2150)

in the School of Computer Science and Software Engineering at

Monash University

Monash University

November, 2002

© Copyright

by

Hayley Cronin

2002

Contents

List of Tables	v
List of Figures	vi
Abstract	vii
Acknowledgments	ix
1 Introduction	1
2 Stochastic Optimisation	3
2.1 Evolutionary Optimisation	3
2.1.1 Genetic Algorithms	4
2.2 Simulated Annealing	4
3 Particle Swarm Optimisation (PSO)	6
3.1 The Original Model	6
3.2 Improvements	8
3.3 Parameter Determination	10
3.4 Hybrid Models	11
3.5 PSO in a Dynamic Environment	12
4 Comparison of Optimisation Methods	14

5	Evaluation of Optimisation Methods	16
5.1	Problems with Previously Reported Experiments	19
6	Investigation and Improvement of PSO	20
6.1	Testing and Comparison of PSO	20
6.1.1	Problems With PSO	21
6.2	Improvements	22
6.2.1	Neighbourhood	22
6.2.2	Random Movement	23
6.2.3	Step Width	24
6.2.4	Velocity Update	24
6.2.5	The New Model	25
6.3	Evaluation of the New Model	26
6.3.1	Baseline Tests	26
6.3.2	Benchmark Comparison	27
6.3.3	Problems With the New Model	28
6.4	Step Width	28
6.4.1	Adaptive Step Width	29
7	Conclusions and Future Work	32
7.1	Future Work	32
	References	34
	Appendix A: Test Functions	38
	Appendix B: Parameter Sets	41

List of Tables

6.1	Summary of results of tests for the basic PSO model and random sampling on the Alpine function	21
6.2	Summary of results of tests for the improved model on the Alpine function	26
6.3	Summary of results of tests of improved model on benchmark functions . .	27
6.4	Summary of results of tests on benchmark functions with step width increased	29
6.5	Results of tests on benchmark functions with adaptive step width	30
6.6	Results of tests on benchmark functions for the new model compared to Genocop3 and VFSR	31

List of Figures

4.1	Random components of movement for PSO and SA	15
6.1	Influences on particles caught between the global and their personal best positions	22
1	Contour plot of the Alpine Function	38
2	The Rosenbrock Function	39
3	Schaffer's F6 Function	39
4	The Griewank Function	40
5	The Rastrigin Function	40

Particle Swarm Optimisation

Hayley Cronin, BSE(Hons)
Monash University, 2002

Supervisor: Dr. Bernd Meyer

Abstract

Particle Swarm Optimisation is a relatively new method used for the optimisation of non-linear functions. Since the introduction of this method it has been improved by several researchers. It has also been found to be a promising method for optimisation in a dynamic environment, where there is a smooth and gradual change in the landscape. In spite of the improvements introduced and the enhanced performance achieved, Particle Swarm Optimisation is still unable to compete with other more established optimisation techniques, such as Genetic Algorithms and Simulated Annealing. Despite this and due to the potential of Particle Swarm Optimisation for optimisation in a dynamic environment, it is still a method worthy of further investigation and development. In particular, if this method were able to perform within a competitive range of some of the best known optimisation methods it could be of great use in optimising a dynamic environment. This project therefore aims at further improvement to Particle Swarm Optimisation so that its performance is comparable to that of other optimisation techniques, making it a viable alternative for nonlinear optimisation problems. To do this, several models of Particle Swarm Optimisation are tested, with the effects of many of the components of the algorithm being investigated. Through visual observations and analysis of the numerical results of these tests several changes are proposed and found to improve the performance of Particle Swarm Optimisation. This improvement is to a degree that performance of this method is now comparable to that of one of the best known implementations of Genetic Algorithms and other successful methods for four common benchmark problems.

Particle Swarm Optimisation

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Hayley Cronin
November 5, 2002

Acknowledgments

I would like to acknowledge the assistance of Bernd Meyer whose guidance made this project possible. I would also like to thank my family for their patience and my friends and fellow Honours students for their constant moral support throughout this year.

Hayley Cronin

Monash University

November 2002

Chapter 1

Introduction

Particle Swarm Optimisation [KE95] is a relatively new method for the global optimisation of nonlinear functions. There are many related, but quite different, techniques commonly used for this purpose. These techniques include Genetic Algorithms [PK00] which have generally been very successful in the form of the Genocop program [MLS94, MJ96, MNM96, MJ96], and Simulated Annealing which has been found to be an effective optimisation method, particularly in the form of Very Fast Simulated Reannealing (VFSR) [IR]. So far Particle Swarm Optimisation is unable to compete with Genetic Algorithms and other more traditional optimisation techniques in terms of performance for most optimisation problems. This method does however, appear to be a promising alternative for use in a dynamic environment, where there is a changing landscape and non-stationary optimum.

Due to its relative novelty, Particle Swarm Optimisation has not been fully developed and evaluated, although several modifications have previously been made to the basic algorithm, resulting in an overall improvement in performance. These modifications have included the use of neighbourhoods to group particles for communication [EK95] and various schemes for implementing this neighbourhood [Ken99]. Other improvements have focused on the velocity of particles, introducing parameters to limit the velocity and through this balance the search between a more local or global search of the relevant space [EK95, Cle99, SE99, SE01]. Suggestions to limit velocities depending on the size of the search space have also been made [ES00]. Work in this area has also included the combining of Particle Swarm Optimisation with Genetic Algorithms to produce hybrid optimisation techniques [LRK01, Ang98] and the investigation of the performance of Particle Swarm Optimisation in a dynamic environment [CD00, CD01b, ES01a]. Despite the research into improving this method the

performance of Particle Swarm Optimisation is yet to be compared to that of other optimisation methods in order to ascertain how this method compares to successful optimisation techniques.

This thesis evaluates the performance of a basic Particle Swarm Optimisation model and then investigates improvements to Particle Swarm Optimisation, producing a new model with several new features. The most notable of these features are the increase in the random movement of particles, the introduction of spatially-determined local neighbourhoods and an adaptive “step width”, or size “step” that a particle takes each move. The performance improvement given by the use of these new features is then analysed, showing that with these new additions to the model performance can be brought into a range where it is competitive with several successful optimisation methods for four common benchmark problems.

Chapter 2

Stochastic Optimisation

There are many different stochastic optimisation techniques, or techniques based on probability, which are used to find the optimum solution to nonlinear functions. These methods are often used to optimise functions where the optimum solution cannot be simply calculated and work by sampling the search space to find the best solution. Among some of the more popularly used stochastic optimisation methods are Evolutionary Optimisation techniques (including Genetic Algorithms) and Simulated Annealing. Another, newer, stochastic optimisation method is the focus of this thesis, and is known as Particle Swarm Optimisation.

2.1 Evolutionary Optimisation

The Optimisation of nonlinear functions using evolutionary computation methods is an area that has been of interest to researchers for several decades [MF99]. These methods mimic some of the processes found in nature. The reasoning behind the use of these ideas is that evolution itself is an optimiser and processes which have been found to work in evolution can also be of use in other optimisation problems. There are many Evolutionary Computation methods which all find the optimum solution to a problem through: [SMY02]:

- The use of a population of search points, often known as “individuals”.
- Association of a “fitness” (objective value) with each search point according to its position within the search space.

- Selection of the fittest individuals in the population.
- Combination of properties of individuals to produce a new population based on the old.

2.1.1 Genetic Algorithms

One class of these evolutionary computation methods is known as Genetic Algorithms (GAs). GAs have been extensively researched and documented since they were first invented by Holland in 1975 [PK00], while the concepts behind them have been researched since their earlier introduction in 1973 by Rechenberg (Evolution Strategies [Rec73]). This optimisation method works by first populating a space randomly with search points (commonly referred to as “individuals”), as with all evolutionary computation methods. An objective function to calculate the “fitness” of individuals is then specified to be optimised. When updating the population at each generation this technique mimics some of the processes observed in natural evolution and genetics [Dav91]. These processes include [For96]:

Natural Selection: where only the individuals deemed to be the fittest, or closest to the optimum solution, survive and the rest are discarded.

Breeding/Genetic Crossover: where the characteristics (for example, the position) of two individuals are combined to produce offspring.

Genetic Mutation: where some of an individual’s characteristics are randomly altered.

The result of this process is that a new population is created each generation which progresses toward the optimum solution, so that eventually the entire population will be situated around the best solution found.

2.2 Simulated Annealing

Simulated Annealing [KCGV83] is another stochastic optimisation technique, this time based on an analogy between the annealing of solids and the optimisation of combinatorial optimisation problems. Unlike GAs and Particle Swarm Optimisation however, this process is a single-solution method with only one possible solution rather than a population of them. After this solution has been initialised within the search space the following process then occurs:

- The solution is randomly changed to create a new potential solution within the neighbourhood of the old.
- If the new solution is better than the old (ie. if it has a better objective value) the new solution is accepted and the old discarded. Alternatively, the solution may be accepted in place of the old, based on a probability determined by the size of the change in objective value and parameter known as the “temperature”.
- The “temperature” is decreased. This causes the probability of a solution being accepted which is not better than the previous to decrease over time. This allows a more random search at the beginning of the optimisation process, with a more directed search toward the end.

This process is repeated until some stopping criteria, such as a maximum number of iterations or reaching a specific value or range, is satisfied.

Simulated Annealing has been found to be effective in optimising many different problems and is the basis for several other effective optimisation techniques, such as Very Fast Simulated Reannealing (VFSR) [IR].

Chapter 3

Particle Swarm Optimisation (PSO)

Particle Swarm Optimisation was first invented by Kennedy and Eberhart in 1995, several decades after Genetic Algorithms and Simulated Annealing. As with Genetic Algorithms and Simulated Annealing this particular optimisation method is also based on an analogy, in this case between a simplified social model and the global optimisation of nonlinear functions.

3.1 The Original Model

In 1995 Kennedy and Eberhart introduced a new method for the optimisation of nonlinear functions called Particle Swarm Optimisation (PSO) [KE95]. PSO is related to evolutionary algorithms and similar in some ways to GAs, but while GAs model processes found in natural selection and evolution, PSO is based on a simplified social model. Kennedy and Eberhart's research into this idea originated with the wish to create a social simulation using the flocking of birds as its basis. This research started with the reconstruction of simulations by Reynolds [Rey87], Heppner and Grenander, but branched off and developed through several models to become Particle Swarm Optimisation. The idea behind PSO is that when individuals in a population are influenced by both their own experiences and the experiences of those around them the entire population can benefit from the shared knowledge. This creates a population with both global searching abilities, through the knowledge of the global best position found, as well as more local search abilities, through

the use of the individual best position found in velocity updates. GAs also function with a similar sharing of information within the population, however this information sharing comes in the form of breeding between individuals, creating a new set of individuals at each generation, and is a more complex process than in PSO. GAs and PSO are therefore related techniques, with several similarities and correspondences between processes, however they are also rather different in their operation and outcomes.

In the original PSO model invented by Kennedy and Eberhart a population of particles is created within the search space, each with an initial random position and heading providing the initial velocity. Optimisation of the function then proceeds as following:

Repeat:

For each particle in the population:

Calculate the new velocity vector according to:

$$v_{id} = v_{id} + \phi_1 \times rand() \times (p_{id} - x_{id}) + \phi_2 \times rand() \times (p_{gd} - x_{id}) \quad (3.1)$$

where v_{id} is the particle's velocity, p_{id} is the particle's individual best position, x_{id} is the particle's current position, p_{gd} is the global best position, ϕ_1 and ϕ_2 are positive constants and $rand()$ is a random number.

NewPosition = OldPosition + velocity

Compare objective value of new position to value at previous best position found, if better then update previous best.

Compare objective value of new position to value at global best position found, if better then update global best.

Until stopping criteria (eg. maximum number of iterations) is reached.

Both a particle's previous best position and the current best position from among the entire population influence a particle's new velocity. The extent to which these factors influence the behavior of the particles is dependent on the ratio of ϕ_1 and ϕ_2 . If there is a large ϕ_1 then a more localised search occurs with particles being attracted more toward their own previous best position, with a greater ϕ_2 however a more global search occurs.

3.2 Improvements

The basic PSO method suggested by Kennedy and Eberhart has since been improved by other researchers, including Kennedy and Eberhart themselves. These improvements have included the division of the population into local neighbourhoods and changes to the velocity update formula, including the addition of factors to constrain the velocity of particles. This constraining factor has generally been seen to improve the performance of this method. The division of the population into neighbourhoods has shown mixed results, but could be an area for further investigation as this alteration provides greater distribution of particles within the search space, a feature useful for optimising in a dynamic environment.

A locally-oriented paradigm for Particle Swarm Optimisation was introduced by Eberhart and Kennedy in [EK95] where the performance of this new paradigm and the previous method were compared. In this new paradigm topological neighbourhoods were specified and the position of the best particle within the local neighbourhood (p_{ld}) of a particle used instead of the global best (p_{gd}) in velocity updates for that particle. The use of local neighbourhoods in this way encouraged particles to spread out more within the search space and gave a greater emphasis to random searching of the area. Although the local neighbourhood version is less prone to getting caught in local optima, the global method performed the best in the experiments reported.

Additional alterations to the basic neighbourhood have since been suggested by Kennedy [Ken99] and Suganthan [Sug99], Kennedy investigating the effect of neighbourhood topology on the performance of the algorithm and Suganthan implementing a variable-size neighbourhood. In Kennedy's research, various topologies were imposed on the population, including those equivalent to the previously suggested global and local neighbourhoods. However it was found that the effect of neighbourhood topology depends on the problem being optimised and therefore cannot be used to specify a best topology in general. This is reasonable as the performance of a particular optimisation method depends greatly on the nature of the function being optimised and it logically follows that this would also apply to such features of the optimisation method as neighbourhood specification. Suganthan's creation of a variable neighbourhood size [Sug99], starting with a neighbourhood of a single particle and increasing to encompass the entire population, intended to prevent early convergence at local optima, was reported to have "some advantages" [Sug99] over a constant neighbourhood size; the results of these tests were mixed however and are therefore inconclusive.

From these results it would seem that in general a global neighbourhood provides the best performance, however local neighbourhoods do provide a distribution of the population among the search space which could be useful in optimising dynamic environments.

Eberhart and Shi have also investigated the improvement of the original PSO algorithm, in [SE98a] introducing an inertia factor to the velocity update formula intended to help balance the behavior of the PSO between a global and local search. The velocity update formula after the addition of the inertia weight is:

$$v_{id} = \omega \times v_{id} + \phi_1 \times rand() \times (p_{id} - x_{id}) + \phi_2 \times rand() \times (p_{gd} - x_{id}) \quad (3.2)$$

The value of this factor must be chosen through trial and error or else an educated guess, when an appropriate value was chosen this factor was found to improve the performance of PSO, working best when gradually adjusting the velocity over time. This focuses PSO to a local search toward the end of the run, therefore improving precision at this stage. Despite these promising results testing was only carried out on one test function and for a small number of runs. Results could be vastly different when used with other problems and more thorough testing is therefore required in order to ascertain the true effect of this factor.

This inertia factor was again investigated by Shi and Eberhart [SE99, SE01] who studied the performance of PSO using a constant weight [SE99], linearly decreasing weight [SE99] and fuzzy system for the adjustment of the inertia weight over time [SE01]. The testing reported in these papers was more thorough than in that which originally introduced the weight, with three or more different test functions used and different population sizes, dimensions and maximum number of generations. Shi and Eberhart found that a linearly decreasing inertia weight improved the performance of PSO, although it could cause a lack of global search ability at the end of a run. To solve this problem they proposed the use of a fuzzy system to adapt the inertia weight which was then tested and reported in [SE01]. As inputs to the system they used the current best performance evaluation and the current inertia weight, the output from the system being the new inertia weight. After testing on three functions for different population sizes, dimensions and maximum number of iterations the fuzzy system was found to have similar or better results than PSO with a linearly decreasing inertia weight.

Another modification to the PSO algorithm was suggested by Clerc who introduced a constriction factor to constrain and control velocities [Cle99]. The velocity update formula

using the constriction factor is now:

$$v_{id} = K(v_{id} + \phi_1 \times rand() \times (p_{id} - x_{id}) + \phi_2 \times rand() \times (p_{gd} - x_{id})) \quad (3.3)$$

Clerc reports that when tested on two 2D functions this PSO version was found to work very well, although the full details of tests undertaken on this method are not given so the relevance of these results cannot be ascertained. A larger number of dimensions and test problems are required to properly evaluate this method and its effectiveness, as acknowledged by the authors. Studies undertaken by Eberhart and Shi and reported in [ES00] show that the constriction factor is more effective than the inertia factor in improving the performance of PSO for three out of five of the test functions used in their experiments. With this factor therefore improving performance for only just over 50% of cases tested whether or not this factor should be used in general cannot be determined for these experiments and is likely to be dependent on the type of function being optimised.

This value for the maximum velocity was also investigated by Eberhart and Shi [ES00]. It was determined that when using the constriction factor, a maximum velocity equal to the maximum range, commonly denoted X_{max} is the most appropriate choice. The importance of limiting the size of the velocity was emphasised by the results for two of the test functions, the results for both of these functions when using the constriction factor and extremely large maximum velocity (virtually the same as not having a maximum velocity) were not promising, however restricting the maximum velocity to X_{max} greatly improved on these results. Eberhart and Shi did not test any other values for the maximum velocity or vary the value of X_{max} in their experiments, this suggests that although they achieved an improved performance with these settings, better performance may still be achieved with other untested values.

The changes suggested here have shown some promise for improving the performance of PSO. However, in many cases testing was performed on only a few test functions or else tests were inconclusive. Further thorough testing on more test functions is required to provide conclusive evidence of the effects of these improvements.

3.3 Parameter Determination

As well as proposing modifications to the PSO algorithm researchers have also investigated the effect of the various parameters used in PSO to try and find the most suitable values. Shi

and Eberhart investigated the effects of different inertia weights, including a time varying weight, and the maximum velocity on the performance of PSO [SE98b]. They concluded that the best inertia weight depended on the maximum velocity chosen. They also suggested that when nothing is known to aid in the choice of the maximum velocity limiting it to X_{max} is a good choice and that performance improved when using a time varying inertia weight. Tests were only performed on one function however, Schaffer's F6 function, it is highly likely that the precise values for these factors will differ to those given on different functions and it is also possible that even the general trends may change also.

Carlisle and Dozier also looked at parameter selection, although in this case they were trying to determine the best parameter values for a good all-purpose PSO [CD01a]. For a general purpose PSO Carlisle and Dozier recommended a population size of 30 particles using Clerc's constriction formula with $\phi_1 = 2.8$ and $\phi_2 = 1.3$, $K = \frac{2}{|2-\phi-\sqrt{\phi^2-4\phi}|}$ where $\phi = \phi_1 + \phi_2$, a global neighbourhood updated asynchronously and, if X_{max} is enforced, setting the velocity of particles at X_{max} to zero. Although Carlisle and Dozier report good results for the five functions tested when using these parameter values these five functions are a very limited set to be able to state that this PSO model will be effective for all, or even most, problems. With the large number of possible optimisation problems it would be near impossible to prove any version effective for the majority of these.

3.4 Hybrid Models

Other modifications made by researchers have also been found to be effective in improving the performance of PSO. Among these are hybrid models between the standard GA and PSO models created by Løvbjerg, Rasmussen and Krink, which use the GA features of breeding and subpopulation with the traditional PSO velocity and update rules [LRK01]. In these models fitness was not used for selection of the particles to breed, instead the particles were chosen with a predefined probability. From this group two were then chosen at random and arithmetic crossover of their position performed to produce two children who then replaced their parents in the population. All particles that were not bred remained unchanged. Subpopulations were also introduced to the model, the only interaction between these different subpopulations being through breeding. Breeding between the different subpopulations had a separate probability to breeding within the same population, the optimum probability of breeding between subpopulations was found to be dependent on the number of subpopulations. The use of subpopulations in this way was intended to

maintain the diversity of the population to avoid convergence at a suboptimal point. Results of testing this model were mixed, the hybrid model being out-performed in terms of best optima found by both the standard PSO and GA models on two of the test functions, while performing better on the other two test functions, both in terms of best optima found and speed of convergence.

A hybrid model was also investigated by Angeline, as reported in [Ang98], who used selection with the standard PSO model. In this version the particles were ranked in order of their current fitness and the best half chosen to replace the other half of the population. The best half merely overrode the current position of the less fit particles with their own position, leaving the previous best position unaltered. When tested on four common benchmark functions the performance was found to improve considerably, although a population size of 125, an unusually large number, was used which could have affected the overall success of these results.

3.5 PSO in a Dynamic Environment

Although the average performance of PSO shows that it is not yet likely to become a realistic alternative to more traditional optimisation techniques, it does appear to be a promising method for optimising functions in a dynamic landscape where the optima are not stationary, but moving gradually in a continuous movement. PSO appears a promising method for optimisation in this case due to the wider distribution of the population around local optima in the search space, which would allow optimisation to continue if one local optimum were to increase in objective value, becoming the new global optimum. Preliminary research by Carlisle and Dozier using the velocity update formula developed by Eberhart and Shi [SE99], shows that with slight modifications the PSO algorithm can be effective in both dynamic and static environments [CD00, CD01b]. Carlisle and Dozier felt that by not allowing the swarm to make decisions based on previous experience after a change in the environment it would be capable of optimising the new function. This was achieved by resetting the particles' previous best position, preventing them from performing updates to velocity and position according to outdated information [CD00]. This was done in two ways with both tested separately, by resetting the particles' velocity periodically and when triggered by a change in the environment. The triggered resetting was implemented through randomly allocating a sentry particle which compared its fitness before it moved to the fitness value remembered from the end of the last iteration, notifying the population if a

change had occurred. The modified models were able to find a non-stationary optima when the movement of the optima is constant in both direction and velocity as well as track this optima over time.

Eberhart and Shi have also investigated the effectiveness of PSO at optimising a dynamic problem [ES01b]. In this case they altered the inertia factor of the velocity update formula only, making its value dependent on a random component. This is due to the fact that when the environment is changing it cannot be known whether a large or small inertia weight is most appropriate. If the inertia weight is randomly determined from within a certain range it is hoped that the inertia weight will be appropriate at least some of the time. Even though PSO was successful at tracking the optima testing was only performed for one evaluation function and the authors acknowledge that further investigation is necessary.

Chapter 4

Comparison of Optimisation Methods

There are several similarities between PSO, Genetic Algorithms and Simulated Annealing. Firstly, all of these methods are stochastic techniques, techniques which use probability, or directed randomness, in order to optimise a function. Genetic Algorithms and PSO also both have a population of search points which is updated repeatedly to search the relevant space for the optimum solution. In the case of PSO and GAs there is also communication between the population, so that members of the population benefit from the experiences of the rest of the population. In PSO this communication takes place through the use of the velocity update formula, where the particle updates its velocity based not only on its current velocity and previous best position, but also on the previous best position found from among the entire population. In Genetic Algorithms this communication is far more direct and individuals do not update, but are replaced. These replacement particles are the result of “breeding” between the previous population, where attributes of the individuals (eg. position) are combined to create new individuals.

The concept of local neighbourhoods in PSO is one which also has a parallel in the use of subpopulations in Genetic Algorithms [GW93]. Both subpopulations and neighbourhoods can be implemented in several different ways, but in all cases they are intended to maintain the diversity of the population. This diversity is an important feature in PSO, as it is because of this that PSO appears to be a promising method for optimising in a dynamic environment.

Despite the similarities and parallels between these methods there are still some important differences which make PSO unique. The main difference is in the way in which the population is updated at each iteration. In each of these methods the position of individuals within the search space is changed, however in different ways. Simulated Annealing has a single solution which is moved at random. The new position may or may not be accepted, depending on the objective values of the new and old positions. This results in a population whose fitness will usually improve, it is only according to a specified probability that new positions of lower objective value than the previous will be accepted. In Genetic Algorithms also, combining the positions of two individuals will, due to the inclusion of the selection operation, guarantee that the population will over time improve. This is not the case for PSO, as although particles are directed toward the areas of the search space that have previously proven successful, there may be areas in between a particle's current position and these good areas which have low objective values and in which a particle could find itself. This effect will depend on the objective function being optimised however, and the position of the particle.

Another difference between PSO and SA in particular is the randomness of movement. In PSO there is very little randomness in the movement of a particle. The only randomness found is in the amount that the individual particle's previous best position and the population's previous best position contribute to the new velocity of a particle. In SA there is total random movement which allows for a more thorough search of the relevant space and the ability to avoid getting caught in local optima, something which PSO is prone to. This difference in randomness of movement is shown in figure 4.1 where the PSO particle is seen to update its velocity based on a random proportion of its previous best and global best vectors. The movement of the SA particle on the other hand is entirely random and independent of other vectors.

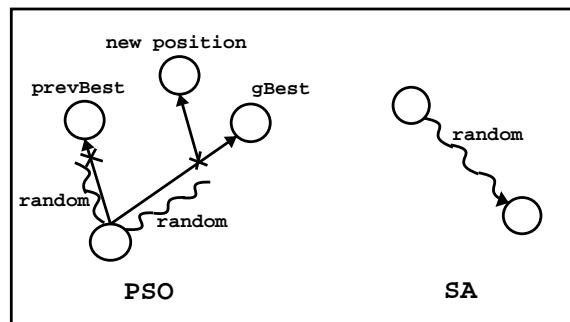


Figure 4.1: Random components of movement for PSO and SA

Chapter 5

Evaluation of Optimisation Methods

In order to measure the performance of optimisation methods tests must be carried out using standard benchmark functions. Once tests have been carried out the performance of an optimisation method is commonly measured in two ways:

- Objective value- This can be measured according to the mean best objective value achieved, the best/worst objective value found of n runs, the percentage of runs resulting in a value within a specified range and the standard deviation maximum error in the best objective values found by n runs.
- Speed of convergence on the optima- The above measurements can be taken for runs of different numbers of iterations or over different runtimes, or else the amount of time taken to find a result close to the best result eventually found for the entire run can be measured.

These values, as well as testing over a large number of problems in order to check the robustness of the method, allow the following important questions relating to performance to be answered [BGK⁺01]:

1. What is the best solution found?
2. How long does it take to determine the best solution?
3. How quickly does the algorithm find a good solution?

4. How robust is the method?

Once these statistics have been gathered a model can easily be compared to other stochastic techniques and programs through published benchmark tests. This comparison could include Genocop, an optimisation program which uses Genetic Algorithms [MLS94, MJ96, MNM96, MJ96] and other successful optimisation programs such as VFSSR [IR]. Care must be taken when selecting benchmarks for comparison however. PSO is intended for unconstrained optimisation, while many other optimisation techniques, including Genocop, are mostly used for problems involving constraints. Benchmark tests which report results for constrained problems cannot be used for comparison with PSO and results from the optimisation of unconstrained problems only must therefore be used.

One appropriate benchmark can be found at [Jan99] and is a comparison of stochastic global optimisation methods. This set of results appears to be a fair comparison of some of the best optimisation programs for unconstrained problems and has therefore been used in this Thesis to gain a fair comparison of performance between PSO and other optimisation methods.

In order to ensure that performance comparisons are fair and accurate there are several other factors to be considered, these are:

Number of Iterations or Runtime: The length of time over which an optimisation program is run will affect the best objective value found in many cases. This is because up until a specific point is reached (which varies depending on the problem and method) continuing to optimise for a longer period of time will continue to find better objective values.

Number of Runs: Although repeating an experiment only a small number of times is likely to demonstrate trends, a large number of repeats of tests is necessary to ensure that tests are reproducible and provide strong statistical evidence on which to base conclusions.

For these reasons all tests reported in this Thesis have been run 100 times and the best, worst and mean result reported, demonstrating the reliability of tests. Tests were also run over 20,000 iterations to be consistent with the tests performed in [Jan99], allowing a fair performance comparison.

Four test problems were chosen from the suite used in [Jan99]. These functions are commonly used in tests on PSO and have all been used in [Ken00, ES00, vdBE01, CD01a]

while subsets of these same functions have been used in many other PSO papers. These functions are also often used in the evaluation of Genetic Algorithms and other benchmark tests. These functions are all intended to be minimised. The equations for these functions, as well as the range and number of dimensions used in the benchmark tests and the tests reported in this Thesis, are given below:

The Rosenbrock Function

$$f_1(x) = \sum_{i=1}^n (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \quad (5.1)$$

where $x = [x_1, x_2, \dots, x_n]$ is an n -dimensional real-valued vector, with n equal to 2, 5 and 10. The range for this function is $-2.048 \leq n \leq 2.048$.

The Generalised Rastrigin Function

$$f_2(x) = \sum_{i=1}^{20} (x_i^2 - 10\cos(2\pi x_i) + 10) \quad (5.2)$$

where $x = [x_1, x_2, \dots, x_{20}]$ is a 20-dimensional real-valued vector. The range for this function is $-600 \leq n \leq 600$.

The Generalised Griewank Function

$$f_3(x) = \frac{1}{4000} \sum_{i=1}^{10} x_i^2 - \prod_{i=1}^{10} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad (5.3)$$

where $x = [x_1, x_2, \dots, x_{10}]$ is an 10-dimensional real-valued vector. The range for this function is $-400 \leq n \leq 400$.

Schaffer's "F6" Function

$$f_4(x, y) = 0.5 - \frac{(\sin\sqrt{x^2 + y^2})^2 - 0.5}{(1.0 + 0.001(x^2 + y^2))^2} \quad (5.4)$$

The range for this function is $-100 \leq n \leq 100$.

5.1 Problems with Previously Reported Experiments

There have been several problems with the previous tests reported in this area. The most important of these problems is the small number of test functions used in experiments. For example, in [SE98b, Cle99, SE98a, CD00, CD01b] only one objective function was used in tests, a number far too small on which to base any general conclusions. Use of so few test functions also brings into question the robustness of the model in question.

There have been other problems with the reported experiments, although not ones so widely spread as the use of only one or two test functions. One of these is the use of different population sizes, various researchers use different population sizes in their experiments. While most researchers use population sizes between 20 and 40, some researchers have used larger population sizes, in one case a population size of over 100. A larger population size will generally provide better (or perhaps just quicker) performance and comparisons between tests with different population sizes are therefore unfair.

Chapter 6

Investigation and Improvement of PSO

6.1 Testing and Comparison of PSO

Due to the insufficient experimental results reported in the previous literature it was considered necessary to first implement and test a basic PSO model. The model implemented was that described in [CD01a] which uses the features of a global neighbourhood and Clerc's constriction factor, resulting in the velocity update formula represented by the following equation:

$$v_{id} = K(v_{id} + \phi_1 \times rand() \times (p_{id} - x_{id}) + \phi_2 \times rand() \times (p_{gd} - x_{id})) \quad (6.1)$$

The parameters used in tests were as follows: $\phi_1 = 2.8$, $\phi_2 = 1.3$ and $K = \frac{2}{|2 - \phi - \sqrt{\phi^2 - 4\phi}|}$, where $\phi = \phi_1 + \phi_2$ as recommended. These features are consistently claimed to be the best by researchers in this field, while the parameters are those suggested in [CD01a].

This model was implemented in the form of a Java program which was then tested on a relatively easy function to optimise known as the Alpine function (see Appendix A). This is defined by the following equation:

$$f(x, y) = \sin(x) \times \sin(y) \times \sqrt{x \times y} \quad (6.2)$$

The search range for these tests was $0 \leq x \leq 3\pi$ and $0 \leq y \leq 3\pi$. Whilst this program was running a random sample was also taken from within the search range with one sample taken for every velocity update performed and the best random sample found recorded. At the end of every run the best value found by Particle Swarm Optimisation was compared to the best result found by the random sampling and the number of times that PSO was better was recorded. All tests were performed with a swarm size of 10 and over 100 runs. The program was allowed to run through 1000 iterations, a value chosen after some experimentation of the model was performed and the best value found was not seen to change substantially after this time.

Test Function	Mean	Best	Worst	Won (%)
PSO	6.90469267	7.885600724	4.527037164	35%
Random Sampling	7.86221079	7.885537552	7.745257758	65%

Table 6.1: Summary of results of tests for the basic PSO model and random sampling on the Alpine function

As can be seen in table 6.1, this model did not perform well, only managing to find a result better than random sampling in 35% of the runs. It is obvious from these results that the model requires improvement, as there is very little point to an optimisation method which is unable to beat a purely random search in even half of the tests.

6.1.1 Problems With PSO

As has been seen this standard version of the PSO model did not perform well, only managing to find a better result than random sampling for 35% of runs. In order to try and discover what problems existed with this model the optimisation process was visualised through the use of a program developed. Through observing the optimisation process in this way the swarm was seen to converge very quickly on one of the local optima; some particles were able to reach this local optima, however the majority of the swarm got caught in a sub-optimal position between the best optima found and another local optimum. These particles were presumably caught between their attraction toward both the global best position found and their own previous best position and unable to move towards either.

It was also observed that if no particle was initialised on the peak containing the global optimum it was very unlikely that any particle would find this global optimum position. This was due to the fact that the only influences on particles are their previous velocity, previous best position and the global best position found. Therefore, unless the particle

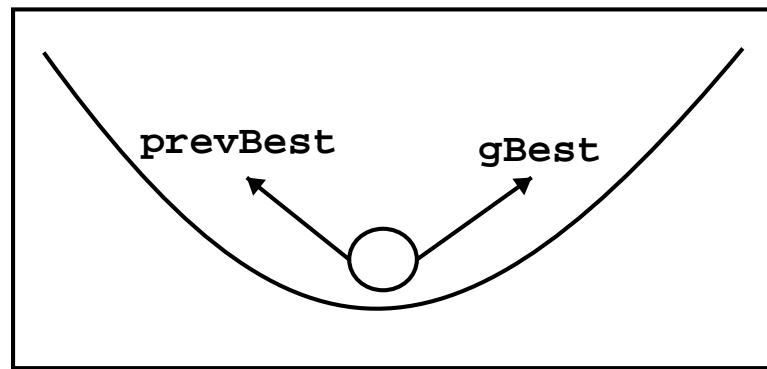


Figure 6.1: Influences on particles caught between the global and their personal best positions

has an initial velocity in the direction of the global optimum, or else the global optimum is between a particle's current position and either it's previous best or the global best positions there will be no movement in the direction of the actual global optimum and this point will not be found. In the case of the experiments using the Alpine Function, the global optimum was situated in the top right-hand corner of the search space and would therefore never be between the particle and either it's previous best or the global best positions. For this reason in these experiments there was almost no chance of the global optimum being found if no particle was initialised on top of the mound containing it. This caused the swarm to often get caught at a suboptimal position. When this occurred it resulted in a very low objective value being found when compared to those found when the swarm was able to find the global optimum.

6.2 Improvements

From the observations of the behavior of the model previously discussed, as well as the investigation of several other optimisation methods and the universal features used by these to assist in optimisation, several areas were seen as possibilities for improvement in the performance of this technique. These ideas were then incorporated into the previous model to form a new one based on the suggestions in [Mey02].

6.2.1 Neighbourhood

The use of local neighbourhoods, instead of a single global neighbourhood, was seen as a possible way to prevent the effect seen in previous experiments where the entire swarm converged on the one local optima. This position may or may not have in fact been the global optimum which meant that the population would often converge at a sub-optimal position. By implementing these local neighbourhoods and making them spatially-determined, clusters of particles could be created which would form around different optima. This would help to maintain the diversity of the swarm in a way similar to how subpopulations are often used in Genetic Algorithms to maintain the diversity of the population. In the case of PSO this diversity would cause a greater distribution of particles within the search space, which should have the added advantage of assisting with optimisation in a dynamic environment. A greater distribution of particles within the search space gives the swarm a better chance of finding a new optimum should the old decrease in objective value, or one of the other local optima increase.

The determination of these spatially-determined neighbourhoods was implemented in such a way that the extent of the influence a particle had on the rest of the swarm was dependent on the objective value of that particle. In order to do this each particle was assigned a “gravity” distance dependent on its objective value and the best objective value found from among the swarm:

$$GravityDistance = MaxGravity \times \max(0.1, 1 - e^{-2(\frac{f(p_{id}) - 0.5 \times f(p_{gd})}{\max(0.01, |f(p_{gd})|)})}) \quad (6.3)$$

When calculating neighbourhoods a particle would be included in the neighbourhood of another if it was within its gravity distance of that particle. In this way particles whose objective value was closer to the best found so far would have a larger gravity distance and greater influence on the swarm than those whose objective value was not as good. The positions of all particles within a neighbourhood were then averaged to give a centre position within the neighbourhood which was then used in the velocity update formula in place of the global best position (p_{gd}) used in previous models.

6.2.2 Random Movement

In the experiments described in the previous section particles were seen to only converge at the global optimum if one of the swarm had been initialised on the peak where that

optimum solution was located. In many cases this resulted in the swarm converging on a suboptimal point, never finding the actual global optimum. Through adding a greater random component to the movement of particles the swarm should be better able to search the entire space and therefore find the global optimum. This idea of randomness is also one found in highly successful optimisation methods such as Genetic Algorithms and Simulated Annealing and is an important factor in both methods.

To add this increased randomness the possibility of a particle making a totally random move, instead of one based on the velocity update formula, was introduced. The probability with which a random movement could be made was a parameter which had to be passed to the program and tuned. An extra random component was also added to the velocity update formula, as described in the “Velocity Update” subsection, this factor added increased randomness to the movement of the particle. However, rather than being an entirely free random movement this randomness is contained within a certain angle, due to the constraints of the rest of the velocity update formula. The addition of this increased randomness in the movement of particles should increase the probability of the global optimum being found, regardless of where particles are initialised in the search space.

6.2.3 Step Width

The “step width” is the distance that a particle will move in one iteration. This is usually determined in the velocity update through the addition of several vectors and is different for each individual particle in the swarm. In this implementation the value of the step width is recalculated each time a velocity update is performed. The step width is determined from the proportion of the objective value of the particle and the best objective value found globally according to the following formula:

$$StepWidth = 1 - e^{-(f(p_{id}) - f(p_{gd}))^2} \quad (6.4)$$

With the step width implemented in this way particles which are very close in objective value to the best found will perform a very fine search of the area, however particles whose objective values are very far from the best found will have larger movements, allowing them to search new areas. Implementing the step width in this way incorporates both local and global searching into the method, depending on the fitness of a particle.

6.2.4 Velocity Update

With these new changes there are now many differences between the new and old velocity update formulas. Now, if a non-random movement is being made a particle's velocity would be updated according to:

$$v_{id} = StepWidth \times norm(w_1 \times norm(v_{id}) + w_2 \times randomHeading + w_3 \times norm(\alpha(p_{id} - x_{id}) + \delta(nHoodCentre - x_{id}))) \quad (6.5)$$

where v_{id} is the particle's velocity, $norm()$ normalises a vector, $nHoodCentre$ is the position of the centre of the particle's local neighbourhood and $w_1, w_2, w_3, \alpha, \delta$ are constant weightings for the various components.

6.2.5 The New Model

Taking into account the improvements just discussed the new PSO model used for the following stage of the investigation proceeds as follows:

Create swarm, initialising each particle with a random position and velocity.

Repeat:

For each particle in the swarm:

Calculate a Step Width based on the formula given previously.

Calculate another step width according to:

$$StepWidth = ParticleSpeed \times e^{-\frac{4 \times Iteration}{MaxIterations}}$$

where $ParticleSpeed$ and $MaxIterations$ are specified parameters.

Select the greater step width to be used.

Generate a random number between 0 and 1.

If number generated is less than the probability of a random move:

$$NewVelocity = StepWidth \times RandomVector$$

Else:

Find the particle's local neighbourhood using the "gravity" distance of particles.

Find the neighbourhood centre.

Update the particle's velocity according to the velocity update formula.

$$NewPosition = CurrentPosition + NewVelocity$$

Until the maximum number of iterations, or some other stopping criteria, is reached.

6.3 Evaluation of the New Model

This new model now contains some of the common features of other optimisation techniques in order to solve the problems found with the previous model. It was now necessary to properly evaluate this model, to determine if these changes were effective in solving the problems found previously and if this model was now competitive when compared to some successful optimisation methods.

6.3.1 Baseline Tests

As with the model of PSO investigated in the previous section the performance of this model was also tested on the Alpine function. Many tests were run with different parameter sets (see Appendix B for full details of these parameter sets and test results) in order to gain an understanding of the effects of these parameters and help determine the best values to be used in subsequent experiments. These tests were all run with a swarm size of 10, the maximum number of iterations equal to 1000 and for 100 runs. The search range for these tests was again $0 \leq x \leq 3\pi$ and $0 \leq y \leq 3\pi$ and as with previous tests a random sample was taken for each velocity update performed and the best result recorded and compared at the end of the run. The results of these tests are shown in table 6.2.

Test Function	Mean	Best	Worst	Beat Random (%)
Equal	7.885039041	7.885583975	7.881041575	96%
More Individual	7.88509437	7.88559921	7.88226118	95%
More Global	7.885019363	7.885596555	7.882618256	97%
No Individual	7.88254403	7.885586419	7.704070381	91%
No Global	7.88503705	7.88559838	7.88268085	93%
No Random	7.885091293	7.885587464	7.883224672	99%
Previous Velocity	7.884669841	7.885596251	7.879513431	98%
More Random	7.885009266	7.885591486	7.882324863	97%

Table 6.2: Summary of results of tests for the improved model on the Alpine function

As can be seen this is a vast improvement over the basic model, beating random sampling in between 91% and 99% of runs of the program. This improvement is significant as this model now functions better than random sampling in most cases. However, it has yet to be seen how the performance of this model compares to that of other optimisation methods, to determine if PSO can in fact be a viable alternative to these and effective for use in a dynamic environment.

6.3.2 Benchmark Comparison

Having determined that this model is an improvement over the previous model it was decided to test this with the four standard benchmark functions described previously. These tests would then allow a fair comparison with other optimisation methods, in particular some of the most successful optimisation programs for these types of problems such as Genocop3 (Genetic Algorithms) and VFSR (Very Fast Simulated Reannealing) through the published results of benchmark tests [Jan99]. All of these tests were performed with a maximum of 20,000 iterations, as was consistent with the tests reported. A swarm size of 20 was used for all tests which were repeated 100 times for each function and the results averaged. The results of these tests are shown in table 6.3.

Test Function	Mean	Best	Worst
Rosenbrock 2D	5.92e-04	3.17e-06	3.15e-03
Rosenbrock 5D	1.50e-01	3.23e-02	3.19e-01
Rosenbrock 10D	3.88e-01	6.18e-01	1.29e-01
Schaffer's F6	2.46e-03	2.46e-03	2.46e-03
Griewank	2.02e+00	4.92e-01	3.60e+00
Rastrigin	4.14e+02	2.53e+02	5.57e+02

Table 6.3: Summary of results of tests of improved model on benchmark functions

When compared to the results of the other methods for these same problems it was seen that this new PSO model now found a solution within a similar range to some of the other methods listed for the Rosenbrock function and a result slightly worse than those found for Schaffer's F6 function. For the Griewank and Rastrigin functions however, this PSO model did not perform well, with the results several orders of magnitude worse than for the other programs. These two functions were therefore chosen as the focus of immediate investigation.

6.3.3 Problems With the New Model

Due to the poor performance of the new PSO model on the Griewank and Rastrigin functions the optimisation of these functions with the new model was investigated further. In order to do this the two functions were first plotted to see if knowledge of the landscape of the functions and knowledge of the optimisation process of the model could gain any insight into the problems with the new model. Through plotting the Griewank function (see Appendix A) it could be seen that the function is the shape of a gradual trough. The surface of this trough is not smooth, but rather made up of many successive steep troughs, with many local maxima and minima. The objective values at two local minima on this trough are therefore very similar and it was reasoned that if there was a particle in one of these local minima and the global best value found was at one of the other local minima, the difference in objective values between these points would be very small. This would result in only a very small step width, which may not be large enough for the particle to move out of this local minima. Particles would therefore get caught in their own local optima and stagnate.

To test this hypothesis the movement of particles during the optimisation process was again observed. During this process particles were seen to begin with some small amount of movement, however this movement slowed down considerably by 7,000 iterations and continued slow down until there was eventually almost no movement at all. Particles came to a near-stop in a position very close to where they began, seeming to support the hypothesis that the inferior performance of this PSO model at optimising the Griewank function, and also the Rastrigin function which has a similarly complex landscape, was due to stagnation of particles.

6.4 Step Width

As it appeared that the size of a particle's step width may have been the cause of problems with the optimisation of the Griewank and Rastrigin functions due to being too small, it was decided to increase the size of this step width and test the effect of this on the performance of this new model for these two functions. To implement this the step width was changed, increasing the size of this factor by a factor of 20. Tests were then run with the same parameters as previously and the results for each function averaged, as shown in table 6.4:

Test Function	Mean	Best	Worst
Rosenbrock 2D	6.63e-01	1.49e-02	2.16e+00
Rosenbrock 5D	4.67e+02	5.49e+01	8.74e+02
Rosenbrock 10D	2.41e+03	8.91e+02	3.68e+03
Schaffer's F6	2.46e-03	2.46e-03	2.46e-03
Griewank	7.12e-02	1.80e-02	1.68e-01
Rastrigin	1.30e+02	1.07e+02	1.50e+02

Table 6.4: Summary of results of tests on benchmark functions with step width increased

As can be seen this change caused a large improvement in the Griewank function and improvement also in the Rastrigin function. The average best value found for the Rastrigin function is in this case actually better than those found by the other methods reported in [Jan99]. This seems to support the theory that it was this variable which was the cause of problems with the optimisation of these problems previously.

As can be seen from comparing the results for the Rosenbrock function to those found previously however, this modification results in a decrease in performance for this function. This decrease in performance took the results for this method out of the range of those found for the other methods reported in the benchmark tests, so that the model is no longer competitive for this function.

Through these results, it was concluded that the new PSO model is able to optimise the Rosenbrock function more effectively with a smaller, constrained, step width and the Griewank and Rastrigin function with a larger step width, while for Schaffer's F6 function this seems to make little difference. It is reasoned that this is due to the nature of the functions - the Rosenbrock function is relatively smooth and easy to optimise, therefore a more refined search of the promising areas works best; however the Griewank and Rastrigin functions are more rugged functions, with many local optima in which particles can stagnate, and a more thorough search of the area to be optimised prevents this stagnation from occurring and allows particles to hopefully find a result closer to the optimum.

6.4.1 Adaptive Step Width

As it is impractical to have two models for use on different problems, particularly since in optimisation the nature of the problem is not always known, it was decided to try and introduce a self-adjusting step width into the PSO model. During optimisation this step width adjusts to a size which is more suitable.

To create this adaptive step width a new factor was added to the step width formula. The rest of the formula was kept the same as previously, so that it was still dependent on the objective value of the particle relative to the best objective value found so far, however now this formula is multiplied by another factor. This factor (“step weight”) is different for each particle in the swarm and begins with a value of 1 so that initially it has no effect. This value is then updated after each movement made by the particle. If the particle has not improved its previous best objective value the step weight is updated according to:

$$StepWeight = StepWeight + 0.00005(xmax - StepWeight) \quad (6.6)$$

where $xmax$ is half the size of the range. In this way the step weight will approach this $xmax$ value but never reach it. If, however, the particle has improved its previous best then the particle is not stagnating and the step weight is reset to 1, so that once again it has no effect on the step width. This means that as the particle does not improve (ie. when it stagnates), the size of the step weight, and therefore the step width, increases giving the particle ever-increasing global search abilities. This continues until the particle finds a position better than any previously found, at which point resetting the step width allows for smaller movements, enabling the particle to thoroughly search the area until it stagnates again or the program finishes.

This new factor is intended to solve the problem of stagnation for the Griewank and Rastrigin functions, while not adversely affecting the performance of the model at optimising the Rosenbrock function and Schaffer’s F6 function. When optimising these functions the particles do not usually stagnate, retaining the more localised search for these functions.

The results for this new model, with the same parameters as previously, when tested on the four benchmark functions are shown in table 6.5:

Test Function	Mean	Best	Worst
Rosenbrock 2D	4.24e-04	1.74e-05	2.06e-03
Rosenbrock 5D	1.67e-01	3.17e-02	4.04e-01
Rosenbrock 10D	5.37e-01	2.45e-01	9.04e-01
Schaffer’s F6	2.46e-03	2.46e-03	2.46e-03
Griewank	2.29e-01	6.09e-02	4.87e-01
Rastrigin	1.67e+02	1.11e+02	2.29e+02

Table 6.5: Results of tests on benchmark functions with adaptive step width

This shows a vast improvement in the performance at optimising the Griewank and Rastrigin functions over previous tests in which the capping on the step width was present, while still keeping similar results for the Rosenbrock function and Schaffer's F6 function. These results can now be compared again to those of the other optimisation programs used in [Jan99] as seen in table 6.6

Test Function	PSO	Genocop3	VFSR	PSO Ranking
Rosenbrock 2D	4.24e-04	0.00e+00	1.64e-11	worst
Rosenbrock 5D	1.67e-01	2.80e+00	3.38e-01	best
Rosenbrock 10D	5.37e-01	8.18e+00	1.47e-01	middle
Schaffer's F6	2.46e-03	0.00e+00	0.00e+00	worst
Griewank	2.29e-01	1.78e-01	6.71e-02	worst
Rastrigin	1.67e+02	8.03e+02	1.91e+03	best

Table 6.6: Results of tests on benchmark functions for the new model compared to Genocop3 and VFSR

This table shows that the performance of this method for the Griewank and Rastrigin functions is now in a similar range to that of the other programs, in the case of the Rastrigin function this performance now beats that of all other programs for which these tests have been reported. As the performance of this method for the Rosenbrock function and Schaffer's F6 function have only altered slightly this performance is also still competitive when compared to these other programs.

Chapter 7

Conclusions and Future Work

The performance of a basic PSO model has been tested and found to be unable to compete with other optimisation methods, or in fact even pure random sampling of the search space for a simple test function. Several improvements have therefore been suggested to this model, to create a new model which addresses the deficiencies seen with the original and incorporates some of the common features of more successful optimisation methods. This model was then altered further, this time with the introduction of an adaptive step width which adjusted the distance that a particle moved at each step according to whether or not it is stagnating. After this addition the performance of this model was again tested and found to be within a competitive range of several of the best optimisation techniques in this area for four common benchmark problems.

7.1 Future Work

This research has tested and evaluated the performance of PSO and introduced a new model which has shown an increased performance for four benchmark problems. Despite these promising results however, four test functions are still only a small number on which to base conclusions. The performance of this model should therefore be tested with a larger number of test functions, in order to be able to properly judge the performance of this new model. This testing could be performed on the remainder of the test functions used in [Jan99] and the performance again compared, to see if the performance increase so far shown by the model is consistent with a wide range of test functions.

Other work could be to again look at the adaptive step width and experiment with other implementations of this. The method used in these tests was only one possible way of implementing this idea and other forms of adaptive step width could be found to be more appropriate. The precise parameters for this step width could also be investigated further to find those which are optimal, either through experimentation or the use of another optimisation method on this parameter. This could also be performed on the other parameters of the method, as these have not been thoroughly tested and the tuning of these could give even better performance.

Now that PSO has been improved to a point where it is competitive with other successful optimisation techniques for several benchmark functions it could also be used to investigate the optimisation of a dynamic environment. This is an important area, as most practical optimisation problems are dynamic in nature. PSO has also appeared to have properties which make it a promising method for use in this case, with the improvements suggested it would be useful to test if this model is still capable of optimising in a dynamic environment. The performance of this method could then be compared to that of other optimisation methods in a dynamic environment, to establish whether this model is, or can be made to be, competitive in this situation also.

References

- [Ang98] Peter Angeline. Using selection to improve particle swarm optimization. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 84–89, Anchorage, May 1998.
- [BGK⁺01] Richard S. Barr, Bruce L. Golden, James Kelly, William R. Stewart, and Mauricio G.C. Resende. Guidelines for designing and reporting on computational experiments with heuristic methods, March 2001. Draft located at <http://www.research.att.com/mgcr/doc/guidelines.pdf>.
- [CD00] Anthony Carlisle and Gerry Dozier. Adapting particle swarm optimization to dynamic environments. In *Proceedings, 2000 ICAI, Las Vegas, NV*, pages 429–434, 2000.
- [CD01a] Anthony Carlisle and Gerry Dozier. An off-the-shelf PSO. In *Proceedings of the Workshop on Particle Swarm Optimization*, Indianapolis, IN, 2001. Purdue School of Engineering and Technology, IUPUI.
- [CD01b] Anthony Carlisle and Gerry Dozier. Tracking changing extrema with particle swarm optimizer, 2001. Auburn University, Technical Report No. CSSE01-08.
- [Cle99] Maurice Clerc. The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization. In *Proc. 1999 Congress on Evolutionary Computation*, pages 1951–1957, Piscataway, NJ, 1999. IEEE Press.
- [Dav91] Lawrence Davis. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.
- [EK95] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43, Piscataway, NJ, 1995. IEEE Press.

- [ES00] Russell Eberhart and Yuhui Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Proc. Congress on Evolutionary Computation 2000*, pages 84–88, San Diego, CA, 2000.
- [ES01a] Russell Eberhart and Yuhui Shi. Particle swarm optimization: Developments, applications and resources. In *Proc. Congress on Evolutionary Computation 2001*, pages 81–86, Piscataway, NJ, 2001. IEEE Press.
- [ES01b] Russell Eberhart and Yuhui Shi. Tracking and optimizing dynamic systems with particle swarms. In *Proc. Congress on Evolutionary Computation 2001*, pages 94–100, Piscataway, NJ, 2001. IEEE Press.
- [For96] Stephanie Forrest. Genetic algorithms. *ACM Computing Surveys*, 28(1), March 1996.
- [GW93] V. Scott Gordon and Darrell Whitley. Serial and parallel genetic algorithms as function optimizers. In Stephanie Forrest, editor, *ICGA-93: The 5th International Conference on Genetic Algorithms*, pages 177 – 183, Urbana-Champaign, 1993. Morgan-Kaufmann.
- [IR] Lester Ingber and Bruce Rosen. Genetic algorithms and very fast simulated re-annealing: A comparison. *Mathematical and Computer Modelling*, 16(11):1990.
- [Jan99] Erich Janka. Vergleich Stochastischer Verfahren zur Globalen Optimierung, 1999. Masters Thesis, University of Vienna, Faculty of Formal and Natural Sciences.
- [KCGV83] S. Kirkpatrick, Jr. C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.
- [KE95] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings IEEE International Conference on Neural Networks*, pages 1942–1948, Piscataway, NJ, 1995. IEEE Service Center.
- [Ken99] James Kennedy. Small worlds and mega-minds: Effects of neighbourhood topology on particle swarm performance. In *Proc. Congress on Evolutionary Computation 1999*, pages 1931–1938, Piscataway, NJ, 1999. IEEE Press.
- [Ken00] James Kennedy. Stereotyping: Improving particle swarm performance with cluster analysis. In *2000 Congress on Evolutionary Computing, Vol. II*, pages 1507–1512, 2000.

- [LRK01] Morten Løvbjerg, Thomas Kiel Rasmussen, and Thiemo Krink. Hybrid particle swarm optimiser with breeding and subpopulations. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, pages 469–476. Morgan Kaufmann, 7-11 2001.
- [Mey02] Bernd Meyer, 2002. Private communication.
- [MF99] Zbigniew Michalewicz and David Fogel. *How To Solve It: Modern Heuristics*. Springer, 1999.
- [MJ96] Zbigniew Michalewicz and Cezary Janikow. Genocop: A genetic algorithm for numerical optimization problems with linear constraints. *Australian Computer Magazine*, December 1996.
- [MLS94] Zbigniew Michalewicz, Thomas Logan, and Swarnalatha Swaminathan. Evolutionary operators for continuous convex parameter spaces. In A.V.Seald and L.J.Fogel, editors, *Proceedings of the 3rd Annual Conference on Evolutionary Programming*, pages 84–97, River Edge, NJ, 1994. World Scientific Publishing.
- [MNM96] Zbigniew Michalewicz, Grish Nazhiyath, and Maciej Michalewicz. A note on usefulness of geometrical crossover for numerical optimization problems. In *Proceedings of the 5th Annual Conference on Evolutionary Programming, San Diego, CA*, pages 305–312, Cambridge, MA, 1996. MIT Press.
- [PK00] D.T. Pham and D. Karaboga. *Intelligent Optimisation Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing and Neural Networks*. Springer, 2000.
- [Rec73] I. Rechenberg. *Evolutionsstrategie- Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, 1973.
- [Rey87] C.W. Reynolds. Flocks, herds and schools: a distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.
- [SE98a] Yuhui Shi and Russell Eberhart. A modified particle swarm optimizer. In *Proceedings of the IEEE International Conference on Evolutionary Computation*, pages 69–73, Piscataway, NJ, 1998. IEEE Press.
- [SE98b] Yuhui Shi and Russell Eberhart. Parameter selection in particle swarm optimization. In *Evolutionary Programming VII*, pages 591–600. Springer, 1998.

- [SE99] Yuhui Shi and Russell Eberhart. Empirical study of particle swarm optimization. In *Proceedings of the 1999 Congress of Evolutionary Computing*, pages 1945–1950, Piscataway, NJ, 1999. IEEE Service Center.
- [SE01] Yuhui Shi and Russell Eberhart. Fuzzy adaptive particle swarm optimization. In *Proc. Congress on Evolutionary Computation 2001*, pages 101–106, Piscataway, NJ, 2001. IEEE Press.
- [SMY02] Ruhul Sarker, Masoud Mohammadian, and Xin Yao, editors. *Evolutionary Optimization*. Kluwer Academic Publishers, Boston/Dordrecht/London, 2002.
- [Sug99] P.N. Suganthan. Particle swarm optimiser with neighbourhood operator. In *Proceedings of the 1999 Congress on Evolutionary Computation*, pages 1958–1962, Piscataway, NJ, 1999. IEEE Press.
- [vdBE01] F. van den Bergh and A. P. Engelbrecht. Effects of swarm size on cooperative particle swarm optimizers. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 892–899, San Francisco, California, USA, 7-11 2001. Morgan Kaufmann.

Appendix A: Test Functions

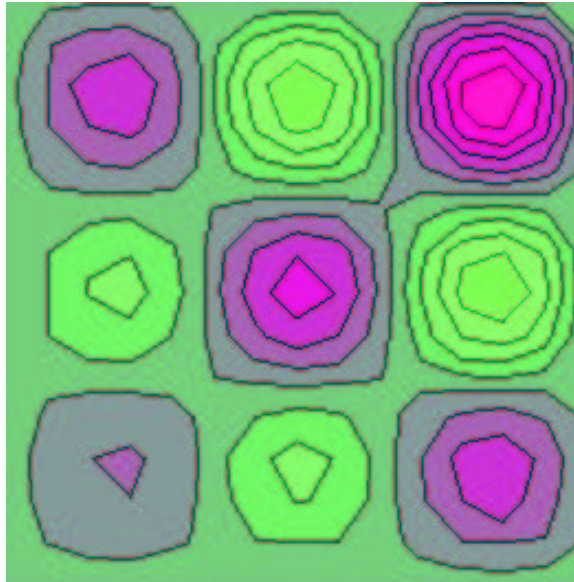


Figure 1: Contour plot of the Alpine Function

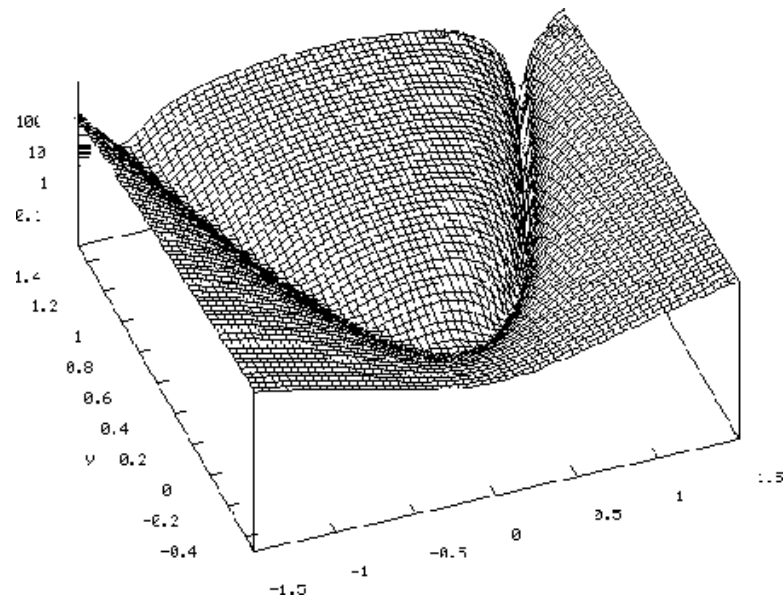


Figure 2: The Rosenbrock Function

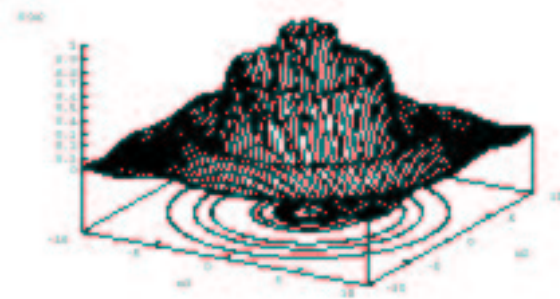


Figure 3: Schaffer's F6 Function

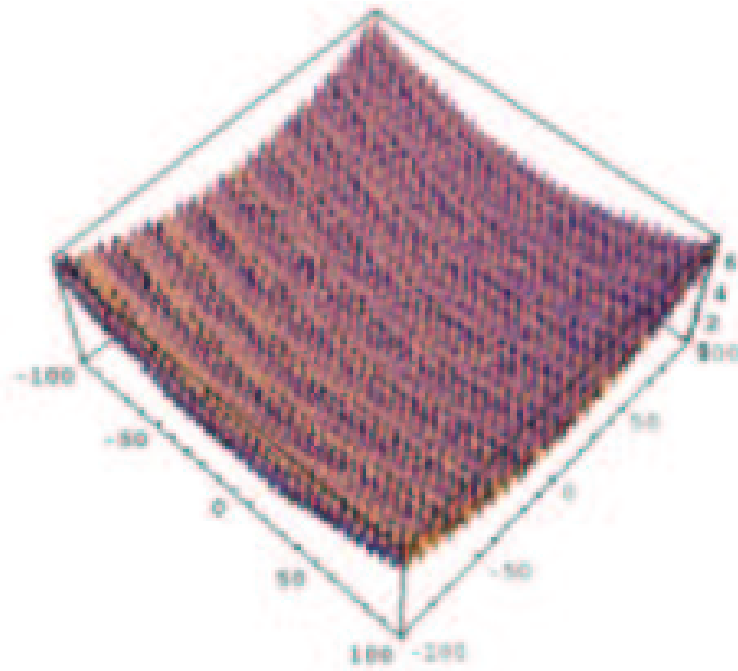


Figure 4: The Griewank Function

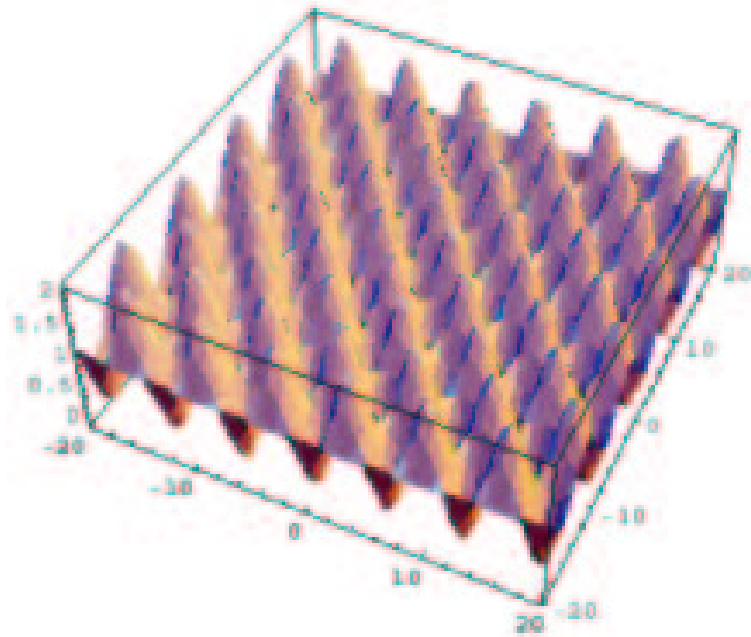


Figure 5: The Rastrigin Function

Appendix B: Parameter Sets

Equal: $w_1 = 1, w_2 = 1, w_3 = 1, \alpha = 1, \delta = 2$

More Individual: $w_1 = 1, w_2 = 2, w_3 = 1, \alpha = 2, \delta = 1$

More Global: $w_1 = 1, w_2 = 2, w_3 = 1, \alpha = 1, \delta = 2$

No Individual: $w_1 = 1, w_2 = 2, w_3 = 1, \alpha = 0, \delta = 2$

No Global: $w_1 = 1, w_2 = 1, w_3 = 1, \alpha = 1, \delta = 0$

No Random: $w_1 = 1, w_2 = 2, w_3 = 0, \alpha = 1, \delta = 2$

Previous Velocity: $w_1 = 2, w_2 = 1, w_3 = 1, \alpha = 1, \delta = 2$

More Random: $w_1 = 1, w_2 = 1, w_3 = 2, \alpha = 1, \delta = 2$