Real-Coded Genetic Algorithm Particle Filters for High-Dimensional State Spaces

Muhammad Shakir Hussain

A dissertation submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy in Computer Science

Department of Computer Science
University College London

Declaration

I, Muhammad Shakir Hussain, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the thesis.

Muhammad Shakir Hussain

Abstract

This thesis successfully addresses the issues faced by particle filters in high-dimensional state-spaces by comparing them with genetic algorithms and then using genetic algorithm theory to address these issues.

Sequential Monte Carlo methods are a class of online posterior density estimation algorithms that are suitable for non-Gaussian and nonlinear environments, however they are known to suffer from particle degeneracy; where the sample of particles becomes too sparse to approximate the posterior accurately. Various techniques have been proposed to address this issue but these techniques fail in high-dimensions.

In this thesis, after a careful comparison between genetic algorithms and particle filters, we posit that genetic algorithm theoretic arguments can be used to explain the working of particle filters. Analysing the working of a particle filter, we note that it is designed similar to a genetic algorithm but does not include recombination. We argue based on the building-block hypothesis that the addition of a recombination operator would be able to address the sample impoverishment phenomenon in higher dimensions. We propose a novel *real-coded genetic algorithm particle filter* (RGAPF) based on these observations and test our hypothesis on the stochastic volatility estimation of financial stocks. The RGAPF successfully scales to higher-dimensions.

To further strengthen our argument that whether building-block-hypothesis-like effects are due to the recombination operator, we compare the RGAPF with a mutation-only particle filter with an adjustable mutation rate that is set to equal the population-to-population variance of the RGAPF. The latter significantly and consistently performs better, indicating that recombination is having a subtle and significant effect that may be theoretically explained by genetic algorithm theory. After two successful attempts at validating our hypothesis we compare the performance of the RGAPF using different real-recombination operators. Observing the behaviour of the RGAPF under these recombination operators we propose a mean-centric recombination operator specifically for high-dimensional particle filtering. This recombination operator is successfully tested and compared with benchmark particle filters and a hybrid CMA-ES particle filter using simulated data and finally on real end-of-day data of the securities making up the FTSE-100 index.

Each experiment is discussed in detail and we conclude with a brief description of the future direction of research.

I dedicate this to my mother and father; amare et sapere vix deo conceditur.

Acknowledgements

I am grateful to my two supervisors, Robert Elliott Smith and Philip Treleaven. This work would not have been possible without their guidance and encouragement. I have benefited a great deal from working closely with Robert. I believe that not only did my research skills improve in the last few years, but he also had a significant effect on my written work and presentation skills. This thesis would not have been possible without his constant guidance, recommendations and revisions.

During the formative years of my PhD, I benefited greatly from the help and guidance of Donald Lawrence. Donald was pivotal in helping me understand the workings of the financial services industry. I would also like to thank Ayub Hanif with whose cooperation I was able to calibrate the Heston stochastic volatility model to the market prices using markov chain Monte Carlo methods. Last but not the least; I would like to thank Fiza Ghaznavi and Adil Hussain for helping me with proof reading and formatting the thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many real-world data analysis tasks involve estimating unknown quantities from some given observations. In most of these applications, prior knowledge about the phenomenon being modelled is available. This knowledge allows the formulation of Bayesian models that use prior distributions for the unknown quantities and likelihood functions relating these quantities to the observations [Jay03, Siv06]. Within this setting all inference on the unknown quantities is based on the posterior distribution obtained from Bayes theorem. Often the observations arrive sequentially in time and one is interested in performing inference on-line. It is therefore necessary to update the posterior distribution as data becomes available [Kal60, Str60].

If data is modelled as a linear Gaussian state-space, it is possible to derive an exact analytical expression to compute the evolving sequence of posterior distributions. The recursion is the well-known and wide-spread Kalman filter [Kal60]. The Kalman filter relies on various assumptions to ensure mathematical tractability. However, real data can be very complex typically involving elements of non-Gaussianity, high-dimensionality and nonlinearity. Many approximation schemes, such as the Extended-Kalman filter [Ju98], Gaussian-sum-approximations filter [KD03] and grid-based filters [AMGC02] have been proposed to surmount this problem. The first two methods fail to take the salient statistical features of the processes under consideration, leading quite often to poor results [Hau12]. Grid-based filters, based on deterministic numerical integration methods, can lead to accurate results, but are difficult to implement and too computationally expensive to be of any practical use [SDFG01].

***Sequential Monte Carlo* (SMC)** methods are a set of simulation based methods which provide a convenient and attractive approach to computing the posterior distributions since no restrictive assumption about the dynamics of the state-space or the density function to be estimated are made [GSS93, SDFG01, AMGC02, Gus10]. SMC methods provide a well-established methodology for generating samples from the required distribution without requiring assumptions about the state-space model or the state distributions. The state-space model can be nonlinear and the initial state and noise distributions can take any form required, however these methods do not perform well when applied to high-dimensional systems [SBBA08, BLA08, Bri11]. SMC methods implement the Bayesian recursion equations directly by using an ensemble based approach. The samples from the distribution are represented by a set of particles; each particle has a weight assigned to it that represents the probability of that particle being sampled from the probability density function. Weight disparity leading to weight collapse is a common issue encountered in these filtering algorithms; however it can be mitigated by including a resampling step before the weights become too uneven. In the resampling step, the particles with negligible weights are replaced by new particles in the proximity of the particles with higher weights [GSS93]. In this thesis the terms SMC and particle filters will be used interchangeably.

There is currently no practical methodology for applying the particle filter in the state estimation of high-dimensional spatial systems [Lee09, BBL08, SBBA08, Bri11]. This research explores methodologies for applying particle filters to high-dimensional state-spaces with the objective of estimating the state distributions with fewer and less restrictive assumptions than the current practical methods. In recent literature the similarities between real-coded genetic algorithms and particle filters have been examined by many researchers [KFZ05, CDD11, SH12a, Hus12]. Based on these similarities, GA theoretic arguments will be used in this thesis to address the causes of collapse of particle filters in high-dimensions.

***Genetic algorithms* (GA)** are population based metaheuristic search and optimization algorithms that mimic the phenomenon of biological evolution. In GAs, the term *chromosome* typically refers to a candidate solution to a problem. The genes are either single bits or short blocks of adjacent bits that encode a particular element of the candidate. An allele in a bit string is either 0 or 1; for larger alphabets more alleles are possible at each locus. The simplest form of a GA involves three types of operators: selection, crossover (recombination) and mutation. The traditional theory of GAs, as formulated in [Hol75], proposes that GAs work by discovering, emphasizing and recombining good *building-blocks* of solutions. The idea here is that good solutions tend to be made up of good building-blocks; combinations of alleles that confer higher-fitness on the strings in which they are present. Holland introduced the notion of *schemas* to formalize the informal notion of building-blocks. These concepts will be discussed in detail in chapter 2.

Real-coded Genetic algorithms (rGAs) use real numbers to represent the genes in the candidate solutions. Encoding is a key issue in GAs since search operators directly manipulate coded representations of the problem and the encoded schema can severely limit the window by which a system observes its world [Koz92]. Fixed length and binary coded strings for the representation solution have dominated GA research since there are theoretical results that show them to be the most effective ones [Gol91] and as they are amenable to simple implementation. For optimization in the continuous domain, it would seem particularly natural to represent the genes directly as real numbers; then a chromosome is a vector of floating point numbers. The size of the chromosomes is kept the same as the length of the vector which is the solution to the problem; in this way each gene represents a variable of the problem. The values of the genes are forced to remain in the interval established by the variables which they represent, so the genetic operators must observe this requirement. Enhanced schema processing is obtained by using alphabets of low-cardinality; however this is a direct contradiction of the results obtained when rGAs were applied in many real world applications [Gol89, Gol91, HH98]. In [Gol91], Goldberg postulated his theory to explain the workings of rGAs. Goldberg showed that his theory is consistent with the theory of schemata and postulated that selection dominates early GA performance and restricts subsequent search to intervals with above-average function-value dimension by dimension. These intervals may be further subdivided on the basis of their attraction under genetic hill climbing. Each of these subintervals is called a virtual character and the collection of characters along a given dimension is called a virtual alphabet. It is the virtual alphabet that is searched during the recombinative phase of the GA; these alphabets are combined via the building-block hypothesis, similar to binary-coded GAs.

Particle filters have a wide variety of applications in the fields of signal processing, finance, target tracking etc. [SDGF01]. In this thesis the dual-estimation of the stochastic volatility of common stock securities and the parameters of the dynamic model is used as a test problem to evaluate the performance of the proposed algorithms.

Stochastic Volatility **(SV)** estimation problem deals with estimating the volatility of financial instruments using stock market data and is an important application of sequential Monte Carlo methods [Jo08]. The famous Black-Scholes model [BS73] was the starting point of a new financial industry and has been a very important pillar of all option trading since. One of its core assumptions is that the volatility of the underlying asset is constant. It was realized early that one has to specify a dynamic on the volatility itself to get closer to the market behaviour [Jo08]. Stochastic volatility models were proposed to model the time varying property of the volatility of the asset. The stochastic volatility is modelled as an unobservable variable of the asset price. The asset price and the volatility process are modelled as a coupled stochastic differential equation w.r.t time given by:

$$dX(t)/X(t) = \sqrt{V(t)}dW_X(t), \tag{1.1}$$

$$dV(t) = k\left(\theta - V(t)\right)\partial t + \varepsilon\sqrt{V(t)}dW_V(t), \tag{1.2}$$

Here *X(t)* represents an asset price process and *V(t)* is the time varying volatility of the asset. *k*, ε and θ are strictly positive constants and W_X and W_V are scalar Brownian motions in some probability measure. It is assumed that $dW_X(t).dW_V(t) = \rho dt.$ Where the correlation ρ is some constant in [-1, 1]. The dynamics of the state-space can be modelled by a hidden Markov model as shown in figure 1-1.

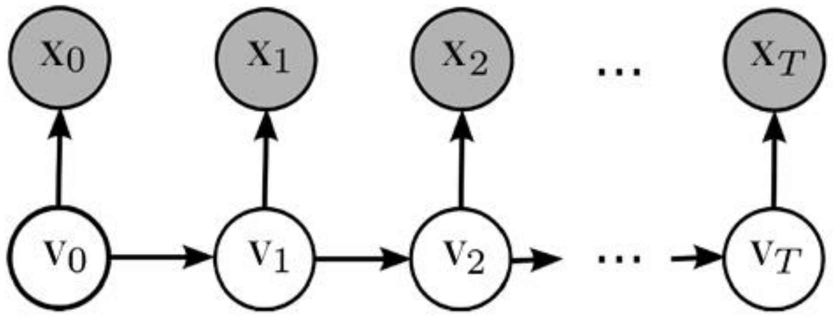


Figure 1-1: The State Space - A Hidden Markov Model

For this dual-estimation problem, Liu et al., in [LW01] proposed a modified version of a particle filter. Their proposed particle filter added random perturbations to the particles in an attempt to address the issue of ensemble collapse. Later in [RB06], Raggi et al., proposed their *particle learning algorithm* (PLA). The PLA was based on the filter of [LW01]. Both these algorithms were able to provide good performance in low-dimensional state spaces, however increasing the dimensions of the state affected their accuracy severely, as will be experimentally demonstrated later in this thesis. These algorithms are discussed in detail in chapter 2.

1.1 Motivation

Estimation and tracking of dynamic systems has been a research focus in statistical mathematics for over five decades [Hau12]. Many estimation methods have been developed that allow statistical estimation for dynamic systems that are linear and Gaussian [Kal60]. In addition, at the cost of increased computational complexity, several methods have shown success in estimation when applied to non-linear Gaussian systems [Gus10]. However, real-world dynamic systems can be both nonlinear and non-Gaussian. The standard Gaussian estimation methods have proven to be inadequate for these problems. SMC methods however are immune to the assumptions of the dynamic model or the process noise and are ideal filtering algorithms for these scenarios. However, these methods do not perform well when applied to high-dimensional systems [BBL08, Bri11, SBBA08]. It has been shown mathematically in [BBL08, SBBA08] that the particle population is required to increase exponentially as the state-dimensions are increased. This requirement is not practically feasible hence there are currently no practical methodologies for applying the particle filter in the state estimation of high-dimensional spatial systems.

Focusing specifically on the SV estimation problem, the filtering algorithms proposed in [LW01] and [RBB06] have proven successful in estimating the model parameters and the stochastic volatility of the asset, however their success is limited to low-dimensions. In [SDGF01], Liu and West, while proposing their version of the dual-estimation filtering algorithm concluded that:

> *"We now have quite effective algorithms for time-varying states as represented throughout this volume. However the need for algorithms that deals with both state and model parameters is specially pressing; we simply do not have access to efficient and effective methods of treating this problem, especially with models with realistically large number of fixed model parameters. It is a very challenging problem." - (Jane Liu & Mike West – Sequential Monte Carlo Methods in Practice)*

The sample impoverishment and the collapse of particle filters in higher-dimensions still remains an open question and to date no successful methodology exists for addressing it.

1.2 Problem statement

The objective of this work is to develop new filtering methodologies that allow SMC filtering methods to be applied to systems with high dimensional parameter spaces with fewer and less restrictive assumptions than the currently practical methods. Reducing these assumptions increases the range of systems that the particle filtering framework can be applied to. The particle filter was developed to meet this objective because restrictive assumptions are fundamental to other filtering methods. The filtering methodologies developed in this thesis would be evaluated on the SV estimation algorithm; however they will not be limited to this specific application area.

1.3 Approach

The issue of ensemble collapse in high-dimensions is addressed in this thesis by exploiting the similarities between particle filtering algorithms and genetic algorithms. The similarities between these two algorithms have been noted by many researchers and various hybrid particle filters have been proposed that utilize population based metaheuristics to optimize the particle population.

The approach of this thesis is rooted in genetic algorithm theory. We hypothesize that if indeed the particle filtering algorithms are similar in design to genetic algorithms, then GA theory can be used to explain the workings of particle filters. Thus we may be able to heuristically identify the reasons behind the collapse of particle filters in high-dimensions. Once these reasons have been identified, we can then address them.

Analysing the particle filter this way led us to the conclusion that a generic particle filter with resampling and regularization is similar to a GA with selection and mutation only. The missing element is a recombination operator. The missing recombination operator and its effect on the working of a GA can be explained by examining the qualitative explanation of the working of a GA given by Mitchell in [Mit98]:

> *"The simple GA increases the number of instances of low-order; short-defining length, high-observed-fitness schemas via the multi-armed-bandit strategy, and these schemas serve as building-blocks* ***<u>that are combined via recombination, into candidate solutions with increasingly higher-order and higher-observed fitness</u>****."*

Translated in particle filtering terminology this would mean that the vector components of the particles that represent the posterior correctly in a particular dimension cannot be combined together on a single string. Hence the efficiency of the particle filter will suffer with an increase in state dimensions. Based on this comparison it is proposed that a recombination operator be added in a particle filter, and a mutation operator with a relatively low mutation rate be used. Thus a *real-coded genetic algorithm particle filter* (RGAPF) is proposed.

To test our hypothesis we carry out an experiment that varies the particle population size for different number of state-dimensions. The RGAPF with arithmetic recombination is run in parallel with two benchmark particle filtering algorithms taken from the SV estimation literature. The results of the experiment show the scalability of the RGAPF to higher-dimensions. The experiment also showed that less number of particles are required by an RGAPF for an accurate prediction; another phenomenon we explained using GA theory.

To further test that recombination is indeed responsible for the building-block like effects; we carry out another experiment comparing the performance of a particle filter with recombination and mutation, with a particle filter with only mutation. A successful outcome of the second experiment further strengthens our belief in our hypothesis that recombination is responsible for building-block like effects, and the combination of these building-blocks is helping the particle filter to scale to higher dimension. We then focus on evaluating different recombination operators within the RGAPF to determine the best possible operator for high-dimensional particle filtering.

1.4 Contribution

The main objective of this research was to address the obstacles to high-dimensional particle filtering and to propose a solution for this limitation. The results of the experiments that were carried out indicate that the proposed particle filters based on genetic-algorithm-theoretic arguments were able to address the dimensional scaling issues faced by particle filters. The results showed the validity of the approach used in this thesis to address this issue.

This thesis makes the following contributions:

- Based on the similarities between particle filters and genetic algorithms it is shown that the schema approach used in GA theory to explain the working of genetic algorithms can be used to analyse the working of a particle filter.
- The real-coded genetic algorithm particle filter (RGAPF) was proposed that addressed the issues encountered by particle filters in high-dimensional state-spaces.
- The thesis experimentally demonstrated the constructive property of a recombination operator compared to mutation in a particle filtering scenario.

- The performance of mean-centric recombination operators is experimentally shown to be superior to other recombination operators in high-dimensional particle filtering.
- The *mean-centric Gaussian recombination* (MCGR) operator is proposed. The MCGR operator provides the best possible estimates of the posterior in the experiments carried out, but is computationally inexpensive compared to other high performing recombination operators in a particle filtering scenario.
- The dual estimation of the stochastic volatility of common stocks and the parameters of the pricing model in high dimensions is addressed successfully.

1.5 Thesis Structure

The next chapter is divided into three sections. The first section gives an introduction to the Bayesian filtering theory, it then goes on to derive the basic sequential Monte Carlo algorithm using the Bayesian recursion equation. The issues faced by particle filters, i.e., of sample impoverishment and curse of dimensionality are then discussed in detail.

The second section begins with an introduction to genetic algorithms and then compares genetic algorithms with particle filters. Based on these similarities it is discussed that GA theory can be used to explain the workings of a particle filter. Using GA theory, the workings of a particle filter is analysed and it is realized that a recombination operator may be able to address the issues faced by particle filters in high dimensions. Based on this observation the real-coded genetic algorithm particle filter is proposed. The final section lays the foundation of the stochastic volatility estimation problem, the test problem of this thesis, and ends with a description of the two particle filtering algorithms that are commercially used for estimating the stochastic volatility and the parameters of the pricing models. These two algorithms will be used as benchmark in this thesis.

The basic setup of all the experiments is similar and is discussed in chapter 4. In chapter 5, the benchmark filtering algorithms are compared with the RGAPF while the particles and the state-dimensions are made to vary. The results of the experiment showed the success of the RGAPG under high-dimensions while using lower number of particles. These results are then explained using GA theory. A further test of our hypothesis was carried out in chapter 6, where a mutation-only particle filter was run in parallel with an RGAPF. The objective of this experiment was to verify whether recombination was in fact responsible for the building-block like effects. The results of this experiment further confirmed the validity of our approach. We then compare the performance of mean centric, parent centric and n-point recombination operators when used in an RGAPF. The results of our experiments show that mean centric recombination operators outperform the other recombination operators in a particle filtering setup. In chapter 7 we revisit our approach, and analyse the results of the experiments carried out in chapter 6. A comparison of the performance of different recombination operators within RGAPF showed the *uni-modal normal distribution crossover* (UNDX) to be providing the best possible estimates of the posterior density. The UNDX operator is computationally expensive to implement and is of quadratic complexity. In chapter 6 we analyse the UNDX operator and propose an alternative *mean-centric Gaussian recombination* (MCGR) operator that provides similar levels of performance but is of linear complexity.

The last chapter of this thesis, chapter 8, discusses the approach, summarizes the experiments and outlines the future direction of research. The thesis concludes with an appendix where the complete results of the experiments are listed.

Chapter 2

Bayesian Filtering, Metaheuristics and the SV Estimation Problem

This chapter reviews background literature that is relevant to the work in this thesis and discusses the test problem used to analyse the performance of our proposed algorithm.

This chapter can be divided into the following main sections:

- In section 2.1, the concepts of Bayesian filtering are introduced and the sequential Monte Carlo methods are discussed in detail. Sequential Monte Carlo methods suffer from the phenomenon of particle collapse, which are then discussed in detail.
- Section 2.2 discusses metaheuristic particle filters that have been successfully used to address a few of the problems that associated with particle filters.
- Metaheuristics are formally introduced in section 2.3, and the field of evolutionary algorithms are then discussed in detail.
- This chapter concludes with an introduction to the stochastic volatility estimation problem that will be used as a test problem later in the thesis.

A brief introduction to various Bayesian filtering methods is given in the next section.

2.1 Bayesian Filtering Methods

Inference methods consist of estimating the current values for a set of parameters based on a set of observations or measurements. The estimation procedure can follow one of two models. The first model assumes that the parameters to be estimated, usually unobservable, are non-random and constant during the observation window but the observations are noisy and thus have random components. The second model assumes that the parameters are random variables that have a prior probability and the observations are noisy as well. When the first model is used for parameter estimation, the procedure is called non-Bayesian or Fisher estimation [LW90]. Parameter estimation using the second model is called Bayesian estimation [Jay03]. Bayesian estimation begins with some initial prior belief; the initial belief statement includes an indication based on some prior probability distribution. Based on the initial belief and a likelihood function describing the event a prediction can be made. The essence of recursive Bayesian estimation is thus:

1. Begin with some prior belief statement.
2. Use the prior belief and a dynamic model to make a prediction.
3. Obtain a posterior belief using the observation model.
4. Declare the posterior belief as the new prior belief and return to 2.

This concept was first formalized in a paper by the Reverend Thomas Bayes, read to the Royal Statistical Society in 1763 by Richard Price several years after Bayes' death.

2.1.1 Bayesian Hierarchy of Estimation Methods

Figure 2-1 shows the hierarchy of Bayesian filters taken from [Hau12] for both non-Gaussian and Gaussian environments. Along the left-hand side are the Gaussian filters and along the right-hand side are all of the Monte Carlo non-Gaussian filters. The main focus in this thesis is on the Monte Carlo filters.

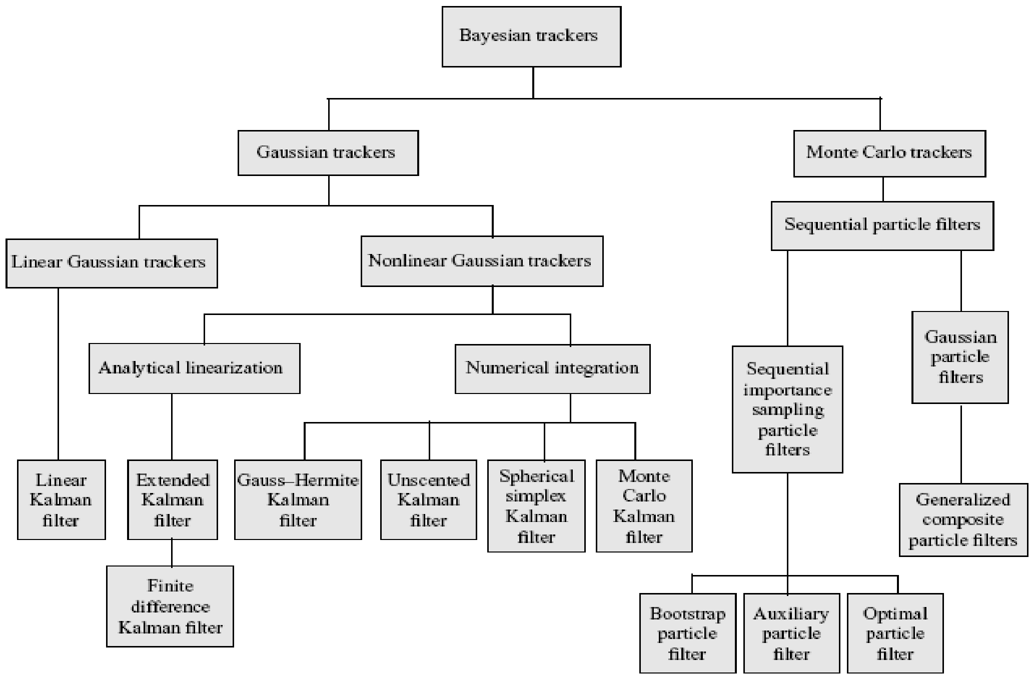

Bayesian trackers
Gaussian trackers
Monte Carlo trackers
Sequential particle filters
Linear Gaussian trackers
Nonlinear Gaussian trackers
Analytical linearization
Numerical integration
Sequential importance sampling particle filters
Gaussian particle filters
Linear Kalman filter
Extended Kalman filter
Gauss–Hermite Kalman filter
Unscented Kalman filter
Spherical simplex Kalman filter
Monte Carlo Kalman filter
Generalized composite particle filters
Finite difference Kalman filter
Bootstrap particle filter
Auxiliary particle filter
Optimal particle filter

Figure 2-1: Hierarchy of Bayesian Filters

2.1.2 General Concepts of Bayesian Estimation

The process of estimation begins with an experiment that provides a set of observable outcomes, usually some form of data. Examples of observable data can include a time-sampled succession of bearings and/or ranges to a target or successive samples of a stock price for sales throughout a day of trading. Based on the observable data, one would like to estimate some characteristic parameters that may be unobservable directly. For example, in case of projectile tracking with bearings-only observations [AMGC02] one would like to estimate the target location and velocity as a function of time. In the case of stock price data, one would like to estimate the volatility of the stock [LW01].

It is always assumed that the parameters to be estimated follow a known recursive dynamic process and that there is a known analytical link between the observed data and the parameters to be estimated. In addition, Bayesian estimation assumes that both the parameters to be estimated and the observed data are stochastic entities. The analytical link between the observed data and the parameters to be estimated provide a unifying framework for estimation where the recursive inference is characterized by a density function for the current state vector value conditioned on the current and all prior observations.

Bayesian estimation has as its objective the estimation of successive values of a parameter vector x given an observation vector z. It is customary to treat both x and z as random vectors. For the parameter vector, the stochastic assumption is inherent in the equations governing the dynamics of the parameter, where un-modelled effects are added as random noise. For the observation vector one can justify a stochastic nature by assuming that there is always some random measurement noise. The random vector x is assumed to have a known prior density function *p(x)*. This prior distribution includes all that is known and unknown about the parameter vector prior to the availability of any observational data. If the true parameter value of x were known, then the probability density of z is given by the conditional density or likelihood function *p(z|x)* and the complete statistical properties of z would be known.

Once an experiment has been conducted and a realization of the random variable z is available, one can use Bayes' law to obtain the posterior conditional density of x:

$$p(x|z) = {p(z|x)p(x)}\big/{p(z)}$$

Thus, within the Bayesian framework, the posterior density contains everything there is to know about x after taking into account the observational outcome of an experiment. Since the experimental outcome z is now available, the denominator of the above equation is just a scalar normalizing constant that can be found from:

$$p(z) = \int p(z|x)p(x)dx$$

For the full Bayesian estimation problem, the likelihood and the posterior, or alternately their joint density, define the statistical model for estimation, where the joint density of the parameter and observational vectors is defined by:

$$p(x, z) = p(x|z)p(z)$$

The solution to the estimation problem is found in the posterior distribution given by the Bayes law. Consequently, the posterior distribution can be used to generate any point estimates of x that are desired, if they exist. Note that the posterior density should be regarded as the most general solution to the estimation problem and in many cases the density function can be used to characterize x.

2.1.3 Introduction to Recursive Bayesian Filtering

A discrete dynamic process will be defined as a process where the current state of the system is dependent on one or more prior states. In continuous processes, the dependence of the current state on previous states is captured within a differential equation. When observations occur at discrete times, estimation conditioned on those observations can only occur at those times, so the differential equation is replaced by its finite difference equivalent that links the state at observation time *t* to states at observation times prior to *t*. A first-order Markov process is one in which the current state is dependent only on the previous state. Thus, we can characterize a discrete random Markov dynamic process as:

$$x_k = f(x_{k-1}, v_{k-1}) \tag{2.1}$$

Here $\{x_k, k \in \boldsymbol{N}\}$ and *f* is a nonlinear function and **N** is the set of natural numbers, $\{v_k, k \in \boldsymbol{N}\}$ is the process noise sequence. The state process is hidden, but we are provided with online measurements of the observation process that is given by the observation equation:

$$y_k = h(x_k, n_k) \tag{2.2}$$

Here $\{n_k, k \in \boldsymbol{N}\}$ is the process noise sequence and h is a non-linear function of x_k. The objective is to recursively estimate x_k whenever we obtain a new measurement of y_k.

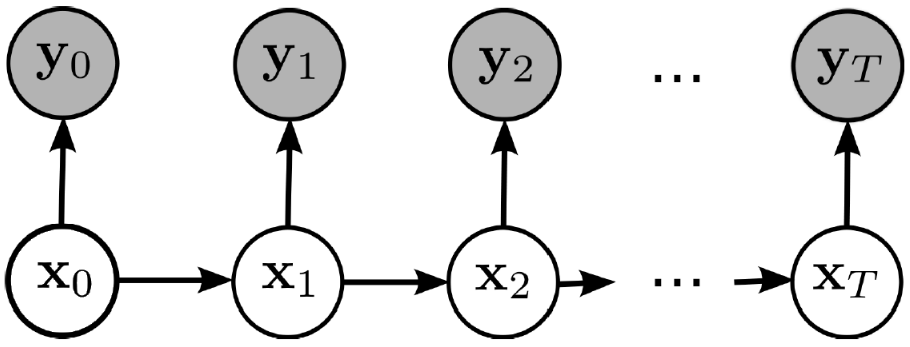


Figure 2-2: The State-Space for Bayesian Filtering

Consider that we know apriori the state and the observation equations, i.e., $x_{k+1} = f(x_k, v_k)$, which can also be assumed to be sampled from $x_{k+1} \sim p(x_{k+1}\,|x_k)$, because of the random noise v_k. Similarly, the observation equation for this hidden process is $y_k = h(x_k, n_k)$. Which is assumed to be sampled from $p(y_k|x_k)$. The values generated by the state equation are hidden and we only have the observation values visible.

In terms of the posterior distribution the Bayes law can be written as:

$$p(x_k|y_{1:k}) = \frac{p(y_{1:k}|x_k)p(x_k)}{p(y_{1:k})} \tag{2.3}$$

$$= \frac{p(y_{1:k-1}, y_k|x_k)p(x_k)}{p(y_k, y_{1:k-1})}$$

$$p(x_k|y_{1:k}) = \frac{p(y_k|x_{k,}y_{1:k-1})p(y_{1:k-1}|x_k)p(x_k)}{p(y_k|y_{1:k-1})p(y_{1:k-1})}$$

$$p(x_k|y_{1:k}) = \frac{p(y_k|x_{k,}y_{1:k-1})p(x_k|y_{1:k-1})p(y_{1:k-1})p(x_k)}{p(y_k|y_{1:k-1})p(y_{1:k-1})p(x_k)}$$

$$p(x_k|y_{1:k}) = \frac{p(y_k|x_{k,})p(x_k|y_{1:k-1})}{p(y_k|y_{1:k-1})} \tag{2.4}$$

One last step is needed to create a completely recursive form for the conditional probability density function equations. The Chapman–Kolmogorov equation provides a link between the prior density, defined as $p(x_k|y_{1:k-1})$, and the previous posterior density.

$$p(x_k|y_{1:k-1}) = \int p(x_k|x_{k-1})p(x_{k-1}|y_{1:k-1})dx_{k-1} \tag{2.5}$$

A single iteration in this Bayesian recursive procedure for developing successive posterior densities is shown in Figure 2-3.

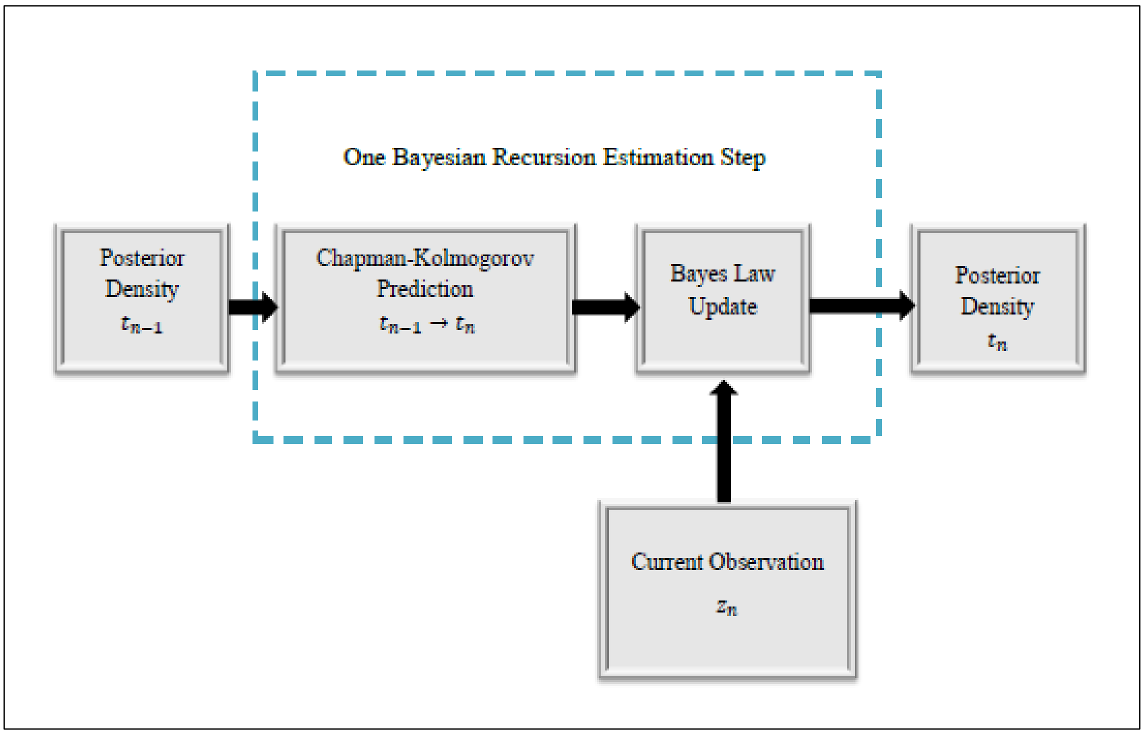

One Bayesian Recursion Estimation Step
Posterior Density t_{n-1}
Chapman-Kolmogorov Prediction $t_{n-1} \rightarrow t_n$
Bayes Law Update
Posterior Density t_n
Current Observation z_n

Figure 2-3: A Single Iteration of Bayesian Filtering

2.1.4 Filtering in a Gaussian State-Space

In the last section, the Bayesian estimation equations were developed using general probability density functions. If the dynamic and observation equations constitute Gaussian processes then under this assumption all of the distribution functions contained in the estimator equations become Gaussian. It is well known that the first two moments of a Gaussian density characterize the density completely [Hau12]. Therefore, a recursive propagation of estimates of the first two moments produces an optimal estimation method for Gaussian processes. These assumptions lead to four classes of Kalman filters:

The Linear Class: When the dynamic and observation equations are both linear and all densities are Gaussian, the integrals can be solved directly leading to the *linear Kalman filter* (LKF).

The Analytical Linearization Class: When all nonlinear functions are expanded in Taylor polynomials and only the linear terms are maintained, the integrals can again be solved directly leading to a Kalman filter form almost identical to that of the LKF. But an additional step requires the computation of the Jacobian of each nonlinear function. These filters consist of the *extended Kalman filter* (EKF) and all of its variants.

The Sigma Point Class: For this class, the nonlinear functions are expanded in more general polynomials such that the integrals reduce to weighted summations over a set of deterministic vector points, called sigma points. Specific filters of this type include the *finite difference Kalman filter* (FDKF), the *unscented Kalman filter* (UKF), the *spherical simplex Kalman filter* (SSKF) and the *Gauss–Hermite Kalman filter* (GHKF).

The Monte Carlo Class: If a set of Monte Carlo samples are drawn from the Gaussian density, creating a discrete density, then the integrals reduce to a sum over discrete random sample points. This method leads to the *Monte Carlo Kalman filter* (MCKF).

2.1.5 Sequential Monte Carlo Methods

Sequential Monte Carlo (SMC) methods are a set of simulation based methods which provide a convenient and attractive approach to computing the posterior distributions [GSS93, SDFG01, AMGC02, Gus10]. Unlike grid-based methods, SMC methods are very flexible, easy to implement, parallelisable and applicable in very general settings [Gus10, SDFG01]. There has been a proliferation of scientific papers on SMC methods and their applications [Gus10] and several closely related algorithms, under the names of bootstrapping filters, particle filters, Monte Carlo filters, interacting particle approximations and survival of the fittest have appeared in several research fields [SDFG01].

In this thesis, the term SMC methods and particle filters will be used interchangeably.

Particle Filter Design

A generic particle filter uses the Bayesian measurement and time update equations to predict the posterior distribution function [AMGC02, Gus10]. The posterior distribution is estimated using N particles $\{x^i\}_{i=1:N}$. Each particle is a potential estimate of the state vector. The particles are assigned weights, w, based on their relative fitness compared to each other. The weights of the particles are then normalized so that they may sum to 1, i.e.:

$$\sum_{i=1}^{N} w_{k|k}^{i} = 1$$

The weight of the particle shows the probability of the particle being drawn from the estimated posterior distribution. A particle which is more probable has more weight compared to other particles in the population. Thus the expected value of the state is the weighted average of the particles in the population. The particles with the greater weight can be considered as having a higher fitness within the population of particles.

The Bayesian filtering equation in case of N particles of a particle filter are modified as follows:

$$\begin{aligned} p(x_{1:k+1}^{i}|y_{1:k}) &= p(x_{k+1}^{i}|x_{1:k}^{i}, y_{1:k})p(x_{1:k}^{i}|y_{1:k}) \\ &= w_{k|k}^{i} p(x_{k+1}^{i}|x_{k}^{i}) \end{aligned} \tag{2.6}$$

Hence:

$$p(x_{1:k+1}\,|y_{1:k}) \approx \textstyle\sum_{i=1}^{N} w_{k|k}^{i} p(x_{k+1}^{i}|x_{k}^{i})\delta(x_k - x_k^{i}) \tag{2.7}$$

Now consider the case where sampling from $p(x_{k+1}|x_k)$ is not computationally feasible, the concept of importance sampling, i.e. generating a sample at random from $x_k^i \sim q(x_k|y_k)$ for each particle and then adjusting the posterior probability for each particle with the importance weight. Hence:

$$\begin{aligned} p(x_{1:k+1}|y_{1:k}) &= \int p(x_{k+1}|x_k)\, p(x_k|y_{1:k})dx_k \\ &= \int q(x_{k+1}|x_k)\frac{p(x_{k+1}|x_k)}{q(x_{k+1}|x_k)}\, p(x_k|y_{1:k})dx_k \end{aligned} \tag{2.8}$$

Thus:

$$p(x_{k+1}|y_{1:k}) = \textstyle\sum_{i=1}^{N} \frac{p(x_{k+1}^{i}|x_{k}^{i})}{q(x_{k+1}^{i}|x_{k}^{i})} w_{k|k}^{i} \delta(x_{1:k+1} - x_{1:k+1}^{i}) \tag{2.9}$$

The solution of the integral appearing the above equation is carried out using Monte Carlo integration using importance sampling [Rub81].. Hence;

$$p(x_k|y_{1:k-1}) = \sum_{i=1}^{N} \frac{p\left(x_k^i \middle| x_{k-1}^i\right)}{q(x_k^i|x_{k-1}^i, y_k)} w_{k-1|k-1}^i \, \delta\left(x_{1:k} - x_{1:k}^i\right) \tag{2.10}$$

Here $w_{k-1|k-1}^i$ is the weight of the i^{th} particle. The estimated value of $p(x_k|y_{1:k-1})$ is then used to update the posterior:

$$p(x_k|y_{1:k}) \propto p(y_k|x_k) w_{k|k-1}^i \tag{2.11}$$

The pseudocode describing the algorithm for a generic particle filtering algorithm, the *sequential importance sampling* particle filter (SIS), is given next.

Algorithm 2.1 : The Sequential Importance Sampling Filter

```
Choose a proposal distribution $q(x_{k+1}^i|x_k^i, y_{k+1})$, resampling strategy and the number of particles N.
Initialization: Generate $x_1^i \sim p_{x_o}, i = 1, \ldots, N$ and let initial weights to be 1/N.
For loop $k$ = 1, 2…End of Observations
START
    1. Measurement Update: For i = 1,2,…, N
           $w_{k|k}^i = \frac{1}{c_k} w_{k|k-1}^i p(y_k|x_k^i)$
       Where the normalization weight is given by:
           $c_k = \sum_{i=1}^N w_{k|k-1}^i \, p(y_k|x_k^i)$
    2. Estimation:
       The filtering density is approximated by
           $p(x_{1:k}|y_{1:k}) = \sum_{i=1}^N w_{k|k}^i \, \delta(x_{1:k+1} - x_{1:k+1}^i)$
       And the mean is approximated by:
           $\dot{x} \approx \sum_{i=1}^N w_{k|k}^i \, x_{1:k}^i$
    3. Time Update:
       Generate predictions according to the proposal distribution
           $x_{k+1}^i \sim q(x_{k+1}|x_k^i, y_{k+1})$
       And compensate for the importance weights
           $w_{k+1|k}^i = w_{k|k}^i \frac{p(x_{k+1}^i|x_k^i)}{q(x_{k+1}^i|x_k^i, y_{k+1})}$
    4. IF ($y_k$ is not last observation)
       $k = k + 1$
       Go to step 1.
         Else End For loop.
END
```

Although the first traces of particle filters date back to the 1950s [HMP54, RR56] and later the control community made some attempts in the 1970s [Han70, AK77], the true particle filtering era started with the introduction of a resampling step in 1993 [GSS93]. The resampling step made particle filter implementation feasible in low dimensional scenarios, but the issue of particle collapse in high dimensions remained a hindrance in its wide spread use [SBBA08, BBL08, QMG08]. A flow chart of the particle filtering algorithm is shown in figure 2-4.

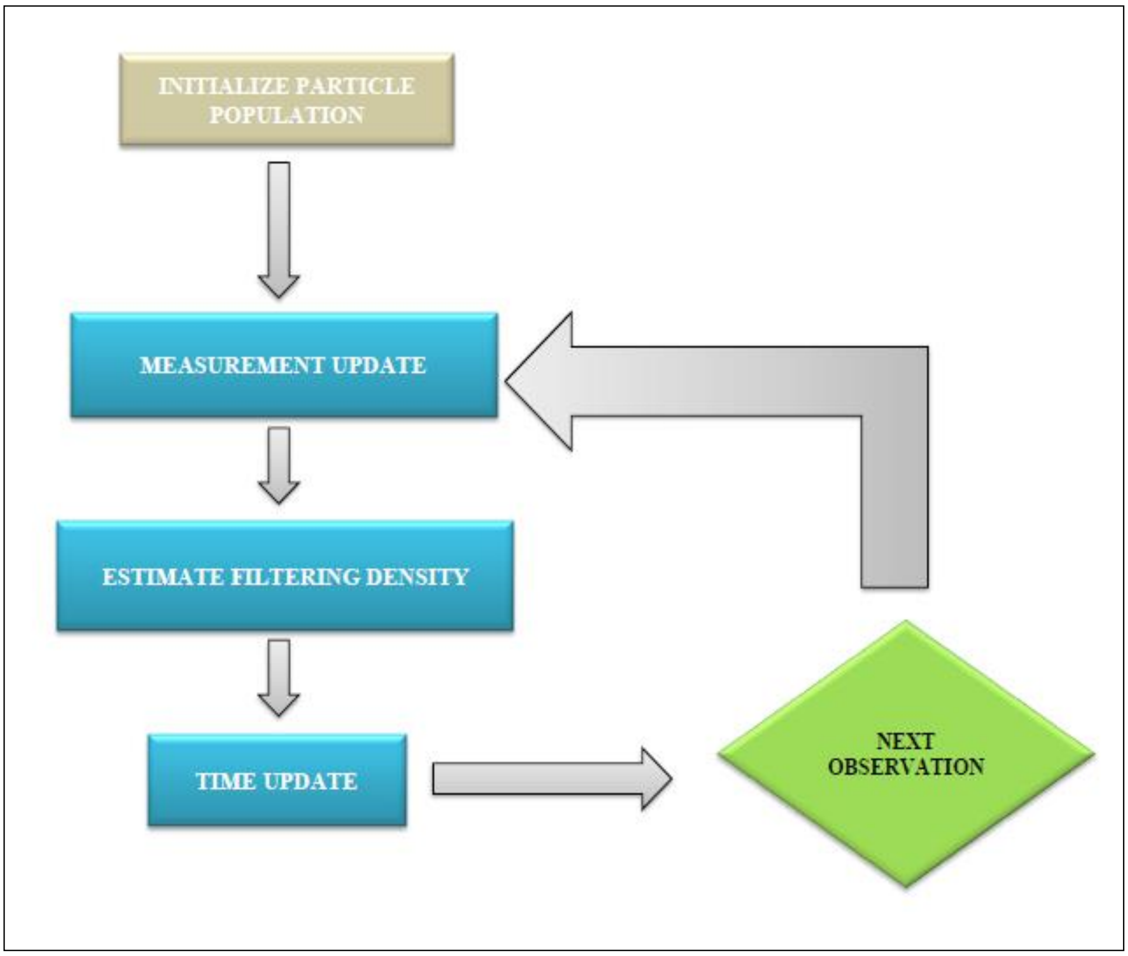


Figure 2-4: The Sequential Importance Sampling Particle Filter

2.1.6 Common Issues in Particle Filters

The phenomenon of the ensemble collapse is known by many names in literature, namely; sample-impoverishment, sample-degeneracy and sample-depletion. Though many different resampling steps have been proposed in literature [AMGC02, RAG04], their main function is to discard particles with negligible weights with particles with above average weights. In low-dimensional cases, they have successfully removed the particle collapse encountered earlier; however they are unable to solve the particle collapse as the dimension of the state increases [SBBA08].

Sample Impoverishment

During the execution of a particle filter, the weights of the particles are updated with the arrival of each observation. However, after a few iterations, the weights of the particles start to be biased towards the particles with a greater weight. Eventually, the particle representation fails and except for a few, all the particles have negligible weights.

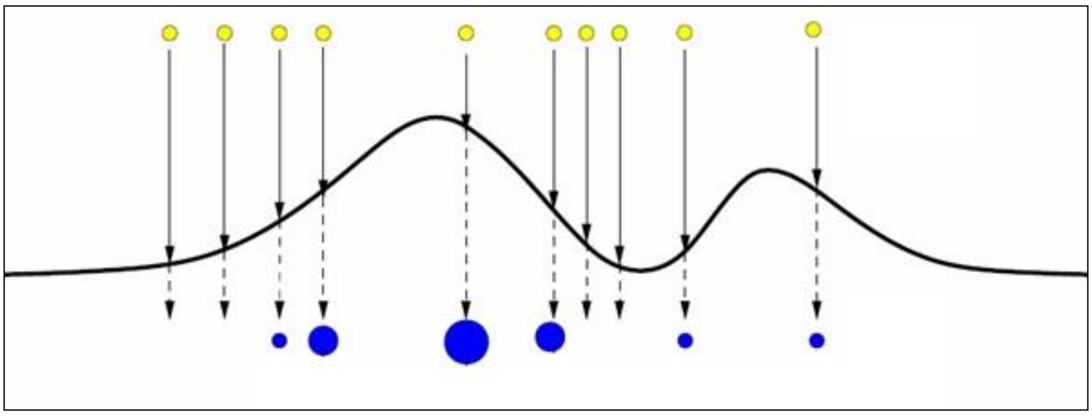


Figure 2-5: Particle Filter – After Initial Iteration

The diagram above taken form [SDFG01] shows the first iteration of a particle filter, where the distribution to be estimated is represented by a set of particles (yellow). After the first iteration, the weights of the particles are updated (blue). The particles with a higher probability are assigned a greater weight, which is shown in the diagram using the size of the blue particles. In this way, a set of particles with their respective weights represent a discrete representation of the probability distribution.

Theoretically, for a particle filter to approximate the posterior density function the weights are required to give a good relative probability of that particle occurring in that distribution. The next diagram below tries to explain the objective of a particle filter. The density function to be estimated is shown by a black line. The yellow circles are the initialized particles that are used to estimate the posterior, and they represent the samples that are drawn from the posterior, the weights of these particles are updated (blue) to represent the probability of that particle being sampled from the posterior. The density function to be estimated can be assumed to be made of an infinite number of particles. The aim is to sample a discrete set of particles form the infinite set that correctly represents the distribution. For an accurate estimate of the posterior, the weights assigned to the particles should be a good representation of the probability of drawing that sample from the actual posterior.

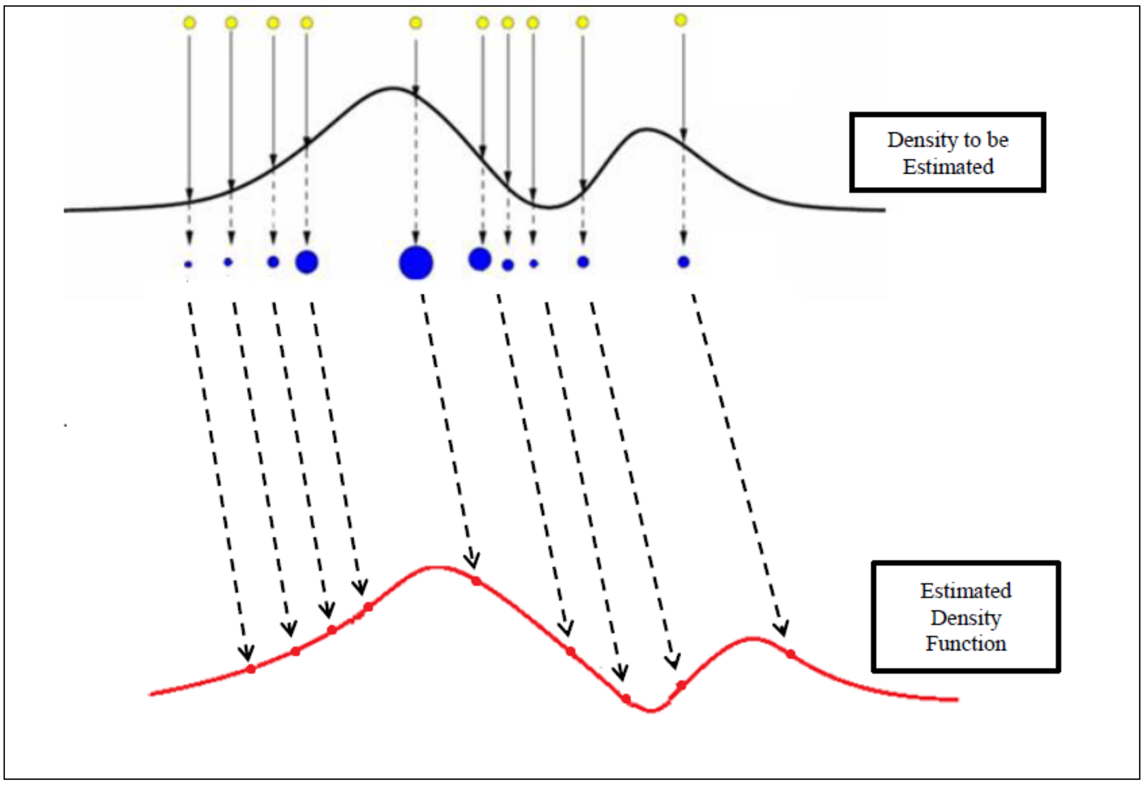


Figure 2-6: Objective of a Particle Filter - Accurate Posterior Density Estimation

The particles and their weights in the above diagram provide an accurate representation of the posterior. However, since in a particle filter the particles are initialized only once, this is only possible if the number of particles approaches infinity. Figure 2-7 shows the phenomenon of ensemble collapse.

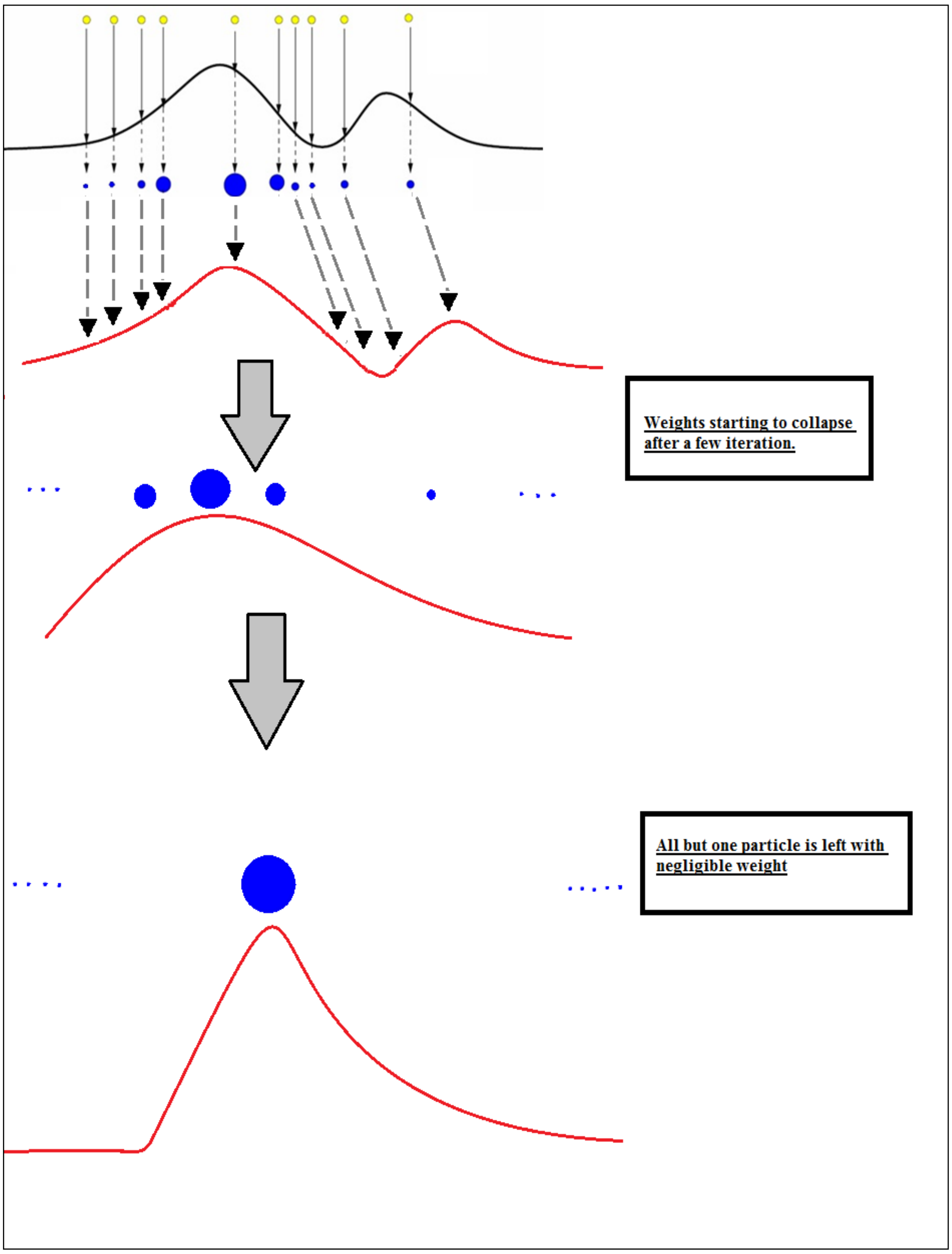

Weights starting to collapse after a few iteration.
All but one particle is left with negligible weight

Figure 2-7: The Phenomenon of Weight Degeneracy

When a particle filter is initialized, all the particles are assigned equal weights. However as the algorithm runs, the weight of the particle with the greatest weight continues to increment and within a few iterations its weight approaches one while all the other particles have negligible weights. Theoretically, this can be avoided by increasing the number of particles, but that is computationally infeasible. Two main methods have been proposed in literature to address this issue. These are discussed next.

Resampling

To address sample impoverishment, Gordon et al., in [GSS93] proposed that the number of particles with above average weights be multiplied within the population by replacing the particles with negligible weights. Adding copies of these particles was able to address the issue in low-dimensions. This technique is called resampling.

The diagram below shows the working of a resampling function. After a threshold is reached, and the weights are biased to a few particles, the particle population is resampled, the particles with negligible weights are replaced and particles with a greater weight are assigned more copies in their neighbourhood space.

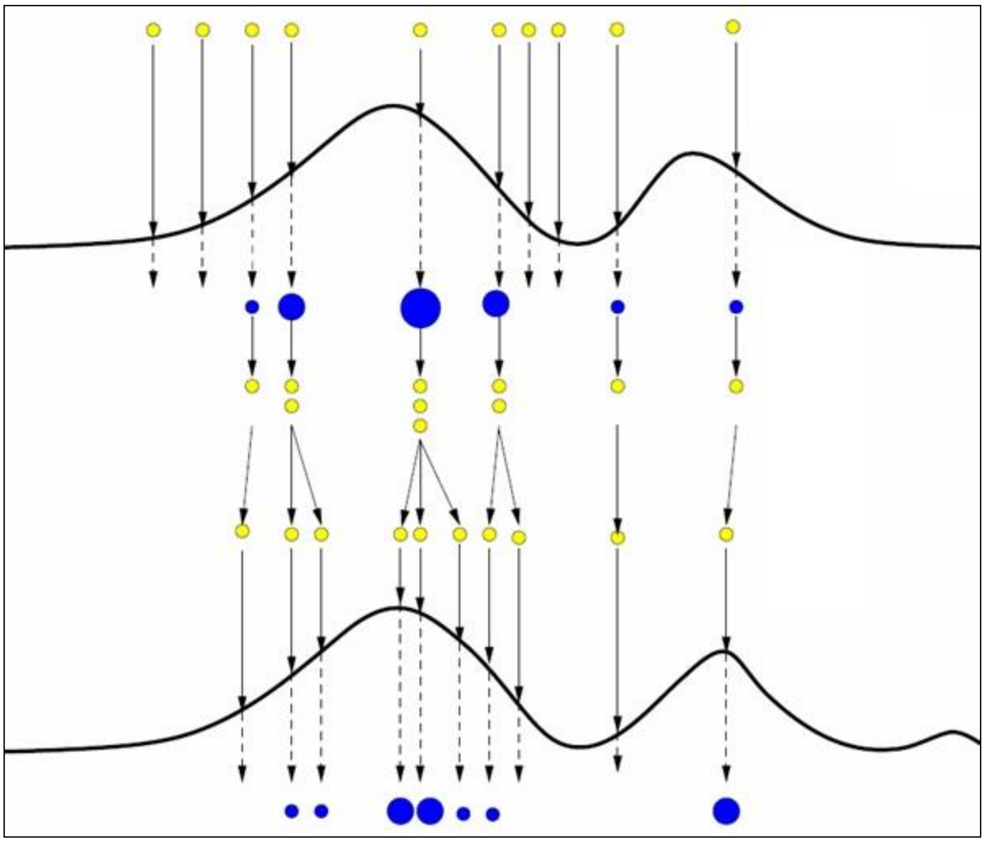


Figure 2-8: Resampling in a Particle Filter

The resampling step, if we were to use terms borrowed from genetic algorithm literature (Genetic algorithms will be discussed later in this chapter), can be seen as a selection operator, and has properties of an exploitation operator, and thus does not explore the search space completely. Because of this, as the number of iterations increase, the same group of particles will be resampled and eventually the whole population of particles will be biased towards a few particles. The particle filter with an added resampling step is called a *sequential importance resampling* particle filter (SIR).

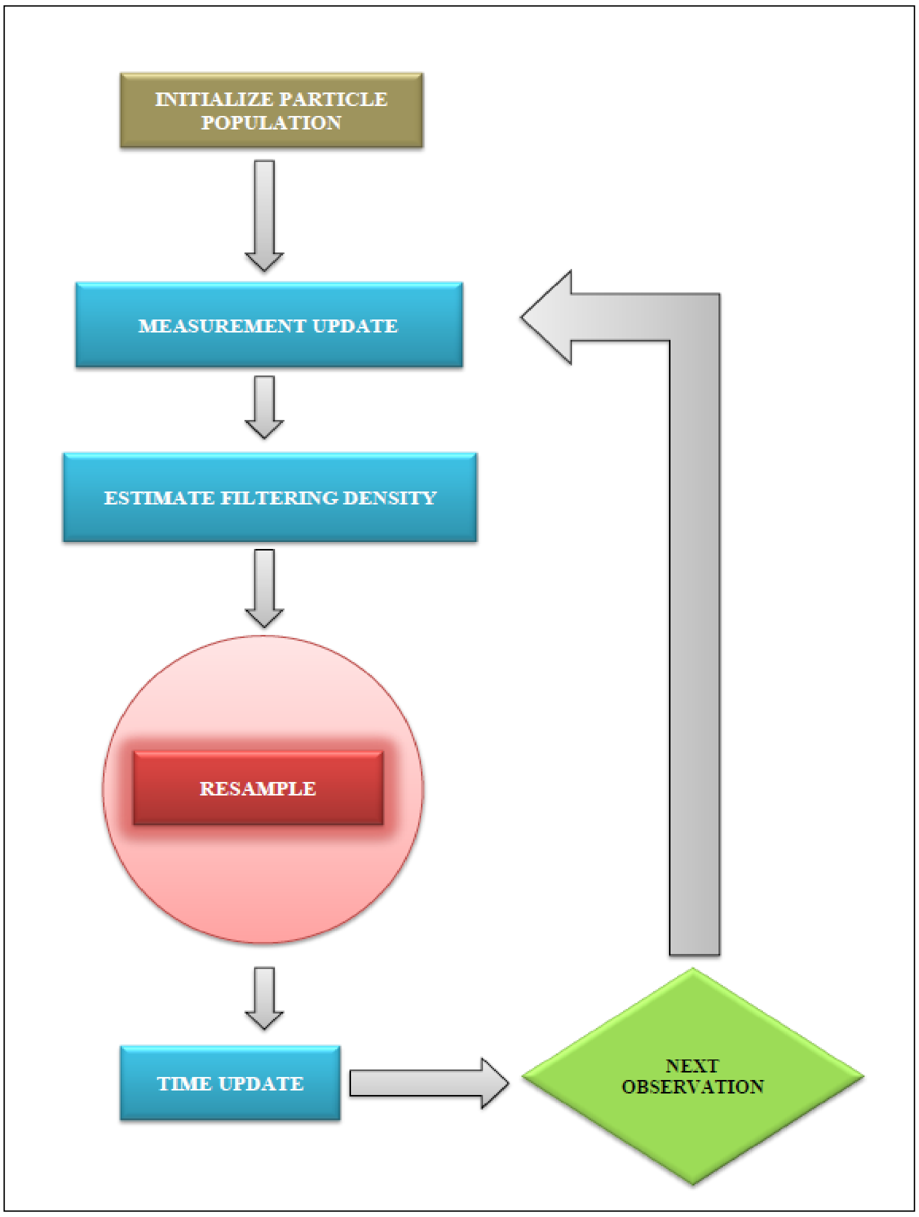


Figure 2-9: Particle Filter with Resampling Step – The SIR Particle Filter

Algorithm 2.2 : The Sequential Importance Re-Sampling Particle Filter

```
Choose a proposal distribution $q(x_{k+1}^i|x_k^i, y_{k+1})$,, resampling strategy and the number of particles N.
Initialization: Generate $x_1^i \sim p_{x_o}, i = 1, \dots, N$ and let initial weights to be 1/N.

For loop $k$ = 1, 2…End of Observations
START
    1. Measurement Update: For i = 1,2,…, N
           $w_{k|k}^i = \frac{1}{c_k} w_{k|k-1}^i p(y_k|x_k^i)$
       Where the normalization weight is given by:
           $c_k = \sum_{i=1}^N w_{k|k-1}^i\, p(y_k|x_k^i)$
    2. Estimation:
       The filtering density is approximated by
           $p(x_{1:k}|y_{1:k}) = \sum_{i=1}^N w_{k|k}^i\, \delta(x_{1:k+1} - x_{1:k+1}^i)$
       And the mean is approximated by:
           $\dot{x} \approx \sum_{i=1}^N w_{k|k}^i\, x_{1:k}^i$
    3. Time Update:
       Generate predictions according to the proposal distribution
           $x_{k+1}^i \sim q(x_{k+1}|x_k^i, y_{k+1})$
       And compensate for the importance weights
           $w_{k+1|k}^i = w_{k|k}^i \frac{p(x_{k+1}^i|x_k^i)}{q(x_{k+1}^i|x_k^i, y_{k+1})}$
    4. IF ($y_k$ is not last observation)
       $k = k + 1$
       Go to step 1.

Else End For loop.

END
```

Regularization and Artificial Evolution

Resampling leads to a loss of diversity among the particles since the resultant sample set will contain many repeated particles for any given weight. To rectify the sample impoverishment due to resampling, after each resampling process a kernel density estimate of the particle density can be used to resample the particles a second time. In this process, each new particle is selected from the resampled particles based on a draw from a uniform distribution and then the sample point is moved a small amount based on a draw from the local kernel. This process tends to concentrate the particles in the region of highest probability and separates them in a random fashion. This method of reducing sample impoverishment is called regularization [Gen92]. A particle filter with resampling and regularization is called a resample and move particle filter. The regularization part constitutes the move part.

There are several alternatives to the resample and move method [Hau11]; including a Markov Chain Monte Carlo (MCMC) sampling method that utilizes the Metropolis–Hastings acceptance algorithm instead of a regularization step and a Gibbs sampling method similar to MCMC. In general, these methods prove to be too computationally intensive for real-time filtering applications [Hau11]. However, in cases where the posterior density has large tail probabilities, as is the case for alpha-stable distributions such as a Levy distribution, the standard SIS particle filter methods may fail due to difficulties in the selection of an appropriate importance density. In such instances, the use of an MCMC method in particle filters provides an alternative for building efficient high dimensional proposal distribution. Other applications where these methods are useful are static parameter estimation and smoothing methods similar to the test problem being addressed in this research [Hua94].

Gordon et al. in [GSS93] introduced a method similar to regularization; they proposed the idea of adding additional random disturbances or roughening penalties to sampled state vectors in an attempt to address the issue of sample degeneracy. They called this method ‘artificial evolution’. Extending this idea to fixed model parameters leads to a synthetic method of generating new sample points for parameters. This ad-hoc idea is similar to using a Gaussian mutation in real coded genetic algorithm literature [ES93].

Consider a state distribution $p(x_t, \theta_t | z_t)$. Where θ_t an estimate of the fixed model parameter, z_t is the observation process and x_t is the hidden state at time *t*.

At time t+1, after the resampling step an independent zero-mean normal increment is added to the parameter.

That is:

$$\theta_{t+1} = \theta_t + \varepsilon_{t+1} \tag{2.14}$$

$$\varepsilon_{t+1} \sim N(\theta, \sigma) \tag{2.15}$$

The key motivating idea is that the artificial evolution provides the mechanism for generating new parameter values at each time step.

Thus the flow chart of the modified algorithm is shown in figure 2.10.

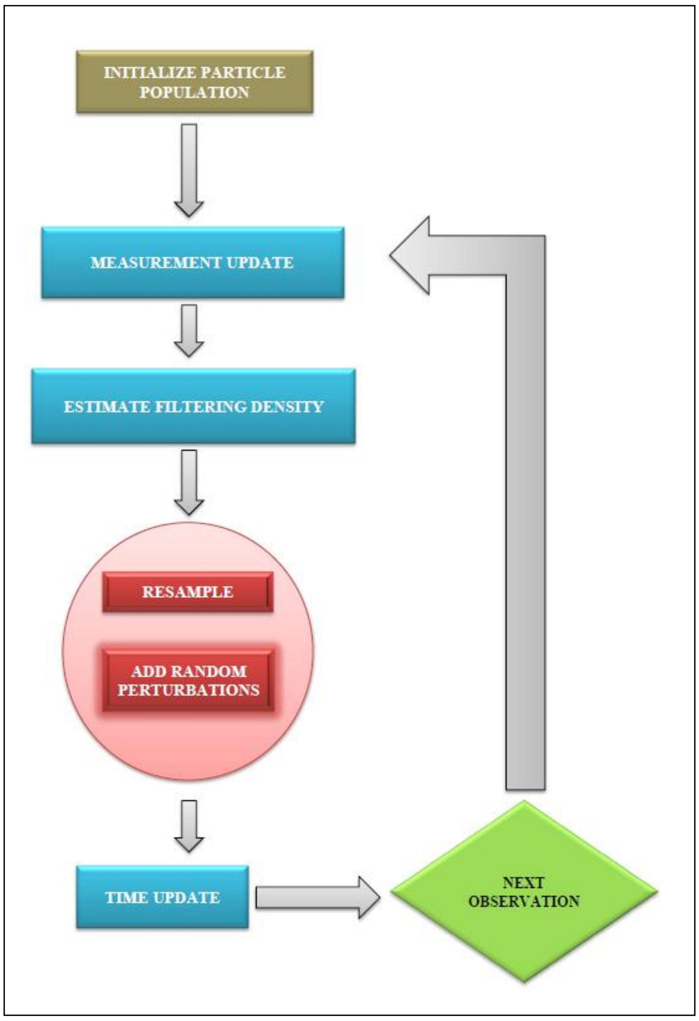


Figure 2-10: Flow chart of a Particle Filter with Artificial Evolution

Algorithm 2.3: The Resample Move Particle Filter

```
Choose a proposal distribution $q(x_{k+1}^i|x_k^i, y_{k+1})$,, resampling strategy and the number of particles N.
Initialization: Generate $x_1^i \sim p_{x_o}, i = 1, \dots, N$ and let initial weights to be 1/N.

For loop $k$ = 1, 2…End of Observations
START
    Measurement Update: For i = 1,2,…, N
        $w_{k|k}^i = \frac{1}{c_k} w_{k|k-1}^i p(y_k|x_k^i)$

    Where the normalization weight is given by:
        $c_k = \sum_{i=1}^N w_{k|k-1}^i\, p(y_k|x_k^i)$

    Estimation:
    The filtering density is approximated by
        $p(x_{1:k}|y_{1:k}) = \sum_{i=1}^N w_{k|k}^i\, \delta(x_{1:k+1} - x_{1:k+1}^i)$
    And the mean is approximated by:
        $\dot{x} \approx \sum_{i=1}^N w_{k|k}^i\, x_{1:k}^i$

    Time Update:
    Generate predictions according to the proposal distribution
        $x_{k+1}^i \sim q(x_{k+1}|x_k^i, y_{k+1})$
    And compensate for the importance weights
        $w_{k+1|k}^i = w_{k|k}^i \frac{p(x_{k+1}^i|x_k^i)}{q(x_{k+1}^i|x_k^i, y_{k+1})}$
    IF ($y_k$ is not last observation)
        $k = k + 1$
    Go to step 1.
        Else End For loop.
END
```

The addition of resampling and regularization were able to address the collapse of particle filters in low-dimensions however as the state-dimensions is increased, the sample impoverishment becomes too severe to be addressed by these methods. In [SBBA08], Snyder et al., showed that the ensemble size required for a successful particle filter scales exponentially with the problem size.

The Curse of Dimensionality

The estimation of continuous density functions using sequential Monte Carlo methods is known to suffer from the 'curse of dimensionality' [And99, BSN03, Lee03]. Snyder et al., in [SBBA08] showed that to avoid ensemble collapse, the particle population needs to increase exponentially with increasing state-dimensions. For a nonlinear estimation problem with zero-mean unit-variance Gaussian noise, they showed that 10^{11} particles are required for a 200-dimensional state-space. Similar observations were reported by Bengtsson et al., in [BBL08].

In [Bri11], Briggs visited the issue of high dimensional particle filtering in state-spaces where the noise distribution is meta-elliptical and the components of the observation vector are independent. The proposed a location-domain particle filter which created a particle population for each component of the observation vector. This greatly increased the space and time complexity of this algorithm. The author noted that compared to the generic particle filter which took 0.034 seconds for an observation update on their test problem, his proposed filter took 2100 seconds. He also noted that with an increase in the number of observation vector components, the time taken by the algorithm for each observation update would increase. This is a significant flaw since for their specific test problem with hundred observations; a generic particle filter took approximately 4 seconds to run, while their proposed location-domain particle filter took approximately 60 hours [Bri11].

Convergence of particle filters

It has been shown in [CD02, Mo98] that given a posterior density function P and a discrete particle population representing this density generated by a particle filter, the following holds true:

$$E[(< P^N, \emptyset >, < P, \emptyset >)^2]^{1/2} \leq C_n \frac{|\emptyset|}{\sqrt{N}} \tag{2.16}$$

Here:

$$< P^N, \emptyset > = \int \emptyset(x)\mu(dx)$$

And $\|\emptyset\| = sup_x \emptyset(x)$, with $\emptyset$ a bounded measureable test function. Thus the equation shows that the RMSE converges to 0 as the number of particles N are increased.

Although it can be argued that using a reasonably large particle population size, one can approach near accurate approximation of the posterior, it was later argued and experimentally demonstrated by Quang et al., in [QMG08] that this equation does not take the dimension of the problem into account. The authors argued that an increase in the dimension of the state can require an exponential increase in the number of particles required for convergence. They showed that the constant C_n changes with changes in the dimension. Later, Snyder et al., in [SBBA08] published a mathematical proof showing that the number of particles required increase exponentially with an increase in state dimension.

2.2 Metaheuristic Particle Filters

The hybridization of particle filters and metaheuristic optimization techniques have been proposed by many authors. In [Pan05], Pantrigo proposed a frame work for combining population based metaheuristics within a particle filter. The author argued that particle filters can be seen as special cases of dynamic optimization being carried out, and based on the population based approach recommended that population based metaheuristics can be added within a particle filter to improve the performance of the algorithm. He used path re-linking and scatter-search within a particle filter, and tested this algorithm on visual articulated tracking of objects with successful results. He proposed that the next frontier for particle filters should be the creation of hybrid-particle filters. Pantrigo's approach was driven by the similarities he observed between dynamic optimization problems in metaheuristics and particle filters.

However Kwok et al., were the first to add a GA inside a particle filter. In [KFZ05] they investigated the sample impoverishment problem in particle filters from the perspective of genetic algorithms. They carried out tests to study the relationship between the number of particles and the time for impoverishment, and concluded that the resampling step is not effective enough to address this issue. They hence proposed a modification to the resampling step and added a simple arithmetic recombination to it and showed experimentally that this addition of an arithmetic recombination inside resampling provided favourable results. They further proposed that mutation can also be added for further research. They tested their proposed approach on a mono-bot simultaneous localization and mapping application. Their main conclusion was that the resampling step should carry out an optimization task and this could lead to better results.

Later Park et al., in [PHRK07] carried out pretty much the same approach proposed by Kwok et al., in [KFZ05]. They also addressed the sample impoverishment phenomenon; however unlike Kwok et al. who added a recombination operator within the resampling step, the authors removed resampling altogether from their proposed algorithm and instead replaced it with a genetic algorithm.

In [ZHM09] inspired by the animal swarm intelligence in the evolutionary computing, the authors proposed a swarm intelligence based particle filter algorithm. The authors argued that unlike the independent particles in the conventional particle filter, the particles in their algorithm cooperated with each other and evolved according to the cognitive effect and social effect in analogy with the cooperative and social aspects of animal populations. Furthermore, they showed that their algorithm is essentially a conventional particle filter with a hierarchical importance sampling process which is guided by the swarm intelligence extracted from the particle configuration. They showed that their modification to the particle filter was able to greatly reduce sample impoverishment in a few scenarios. They compared their proposed approach with several nonlinear filters in the state estimation, and visual tracking.

In [Nak07], Nakano et al., proposed a new filtering technique for sequential data assimilation, the *merging particle filter* (MPF). In the MPF, the filtering procedure was performed based on sampling of a forecast ensemble as in the particle filter. However, unlike the generic particle filter, each member of a filtered ensemble was generated by merging multiple samples from the forecast ensemble such that the mean and covariance of the filtered distribution were approximately preserved. The merging phase was similar to recombination and was shown to address sample impoverishment faced by generic particle filters.

Similarly in [AMZ12], Ahmadia et al., applied yet another metaheuristic within a particle filter. Their paper proposed a new version of the particle filtering (PF) algorithm based on the invasive weed optimization (IWO) method. In order to avert approximation errors due to the initialization of particles, their paper suggests applying the IWO algorithm by translating the sampling step into a nonlinear optimization problem by introducing an appropriate fitness function. The validity of the proposed method was evaluated against three distinct examples: the stochastic volatility estimation problem in finance, the severely nonlinear waste water sludge treatment plant, and the benchmark target tracking on re-entry problem. By simulation analysis and evaluation, it was verified that applying the suggested IWO enhanced PF algorithm (PFIWO) would contribute to significant estimation performance improvements.

The addition of a metaheuristic inside a particle filter was motivated by the population based approach of both the algorithms. All such implementations of hybridization of these algorithms show an improved performance however no investigation into the rationale behind this improvement has been carried out. All of the proposed modified particle filters were christened with different names, yet the underlying gist of their approach was the same:

> *"Adding a metaheuristic layer inside a particle filter provides better results and is also able to improve the performance of the resampling step within a particle filter".*

In [SH12a], [SH12b] and [Hus12], I investigated the reasons behind this improved performance by comparing the SMC methods with genetic algorithms and then using GA theory to explain the obtained results. I argued that the resampling step in SMC methods is similar to the selection operator in genetic algorithms, and the GA theoretic approaches of Schemata and building-blocks can be used to explain the working of particle filters. The experiments carried out in these papers however used a low-dimensional state-space test problem. In this thesis however, high-dimensional state-spaces will be used and the results will be analysed and explained in light of GA theory.

The next section discusses the main concepts of metaheuristic optimization algorithms, and goes on to lay down the foundation of the approach used in this thesis.

2.3 Population Based Metaheuristics

Population-based metaheuristics (P-metaheuristics) are optimization and search algorithms that use a population of candidate solutions and carry out an iterative improvement of the population to search for the best possible solution. After initialization of a population of solutions, a new population of solutions is derived using variation operators. Finally, this new population is integrated into the current one using some selection procedures. The search process is stopped when a given stopping criteria is reached. Algorithms such as *evolutionary algorithms* (EAs), *scatter search* (SS), *estimation of distribution algorithms* (EDAs), *particle swarm optimization* (PSO), *bee colony* (BC) and *artificial immune systems* (AISs) belong to this class of metaheuristics.

2.3.1 Common Concepts of Population-Based Metaheuristics

Most P-metaheuristics are nature-inspired algorithms, however they differ in the way they perform the selection, modification and replacement procedures and the search memory they are using during the search. These steps are described next:

Search memory: The memory of a P-metaheuristic represents the set of information extracted and memorized during the search. In most cases the search memory is limited to the population of solutions.

Generation: In the generation step, a new set of candidate solutions are generated. Based on the class of metaheuristic being used, different operators are available that interact and modify the current population, hence generating candidate solutions in the neighbourhood of the current population.

Selection: The selection step consists in selecting the new solutions from the union of the current population and the generated population. The traditional strategy consists in selecting the generated population as the new population. Other strategies use some *elitism* in the selection phase where they provide the best solutions from the two sets.

2.3.2 Evolutionary Algorithms

Evolutionary algorithms are based on the theory of evolution, proposed by Darwin in 1859, in his famous book *On the Origin of Species*. Different main schools of evolutionary algorithms have evolved independently during the past few decades: *genetic algorithms* mainly developed in Michigan, USA, by Holland [Hol75]; *evolution strategies*, developed in Berlin, Germany, by Rechenberg and Schewefel [Tal09]. Each of these constitutes a different approach; however, they are inspired by the same principles of natural evolution. Evolutionary algorithms are stochastic P-metaheuristics that have been successfully applied to many real and complex problems. Their success in solving difficult optimization problems in various domains has promoted the field known as *evolutionary computation* (EC) [Tal09]. EAs are based on the notion of *competition*. They represent a class of iterative optimization algorithms that simulate the evolution of species. They are based on the evolution of a population of individuals. Initially, this population is usually generated randomly. Every individual in the population is the encoded version of a tentative solution. An objective function associates a fitness value with every individual indicating its suitability to the problem. At each step, individuals are selected to form the parents, following the selection paradigm in which individuals with better fitness are selected with a higher probability. Then, selected individuals are reproduced using variation operators (e.g., crossover, mutation) to generate new off springs. Finally, a replacement scheme is applied to determine which individuals of the population will survive from the off springs and the parents. This iteration represents a generation, as shown in figure 2-11.

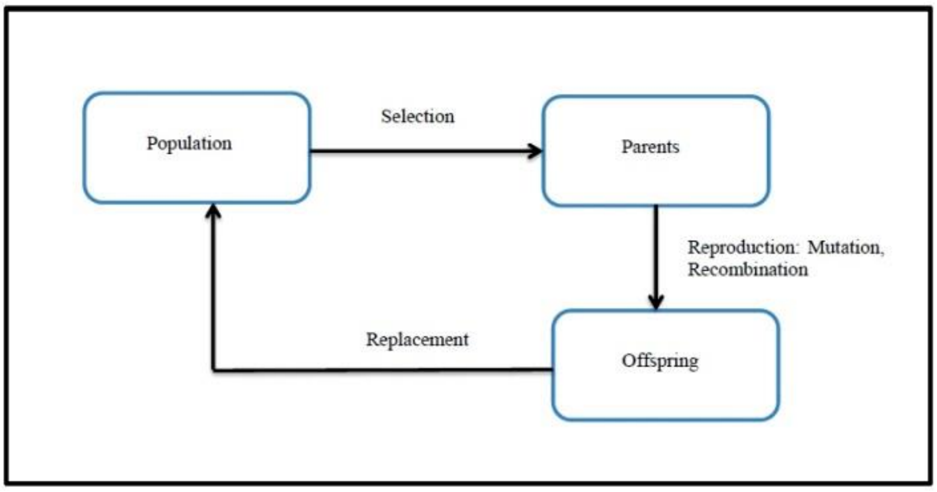


Figure 2-11: Template of an Evolutionary Algorithm

This process is iterated until a stopping criteria hold.

Genetic Algorithms

Genetic algorithms were developed by Holland in the 1960s to understand the adaptive processes of natural systems [Hol75]. A GA uses a crossover and mutation operator to carry out an effective search of the search-space. They use probabilistic selection that samples potential parents from the population, and applies these variation operators on them to generate new candidate solutions in the neighbourhood of the parents. Holland proposed the Schema Theorem to explain the working of a GA. The theoretical foundations of GAs are discussed later in this chapter.

Given a clearly defined problem to be solved and a bit string representation for candidate solutions, a simple GA works as follows:

Algorithm 2.3: A Simple Genetic Algorithm

```
Start with a randomly generated population of $n$ $l$–bit chromosomes (candidate solutions to a problem).
1. Calculate the fitness $f(x)$ of each chromosome $x$ in the population.
2. Repeat the following steps until $n$ offspring have been created:
    a. Select a pair of parent chromosomes from the current population, the probability
       of selection being an increasing function of fitness. Selection is done with
       replacement, meaning that the same chromosome can be selected more than once
       to become a parent.
    b. With probability $p_c$, the crossover rate, carry out recombination on the selected
       parent chromosomes.
    c. Mutate the two offspring at each locus with probability $p_m$, the mutation rate, and
       place the resulting chromosomes in the new population.
3. Replace the current population with the new population.
4. Go to step 2.
```

Each iteration of this process is called a *generation*. The entire set of generations is called a *run*. At the end of a run the fittest population member is the solution to the optimization problem.

Evolution Strategies

Evolution strategies are an important subclasses of evolutionary algorithms. They were originally developed by Rechenberg and Schewefel in 1964 at the Technical University of Berlin [Tal09]. ES are mostly applied to continuous optimization where representations are based on real-valued vectors. Early applications include real valued parameter shape optimization. They usually use an elitist replacement and a Gaussian distributed mutation. Crossover is rarely used. In an ES, there is a distinction between the population of parents of size μ and the population of the off-springs of size $\lambda \geq \mu$. An individual is composed of the float-decision variables plus some other parameters guiding the search. Thus, an ES facilitates a kind of *self-adaptation* by evolving the solution as well as the strategy parameters e.g., mutation step size, at the same time. The selection operator is deterministic and is based on the fitness ranking. Hence, the parameterization of an ES is highly customizable. Their main advantage is their efficiency in terms of time

The basic version of ES, the (1 + 1)-ES, has a population composed of two individuals: the current point (parent) and the result of its mutation (offspring). The parent is replaced by its offspring if it is better; otherwise the offspring is disregarded. More generally, in the (1 + λ)-ES strategy, λ offspring can be generated and compete with the parent. In a (1, λ)-ES the best individual of the λ offspring becomes the parent of the next population while the current parent is always deleted. Most of the recent derivatives of ES use a population of μ parents and also recombination of ρ offspring as an additional operator, which defines the (μ/ρ + λ)-ES strategy where the new population is selected from the parents μ and the offspring λ.

Algorithm 2.4: A Generic ES Algorithm

```
Initialize a population of μ individuals.
1. Calculate the fitness $f(x)$ of each individual, $x$ in the population.
2. Repeat the following steps until stopping criteria is reached:
        • Generate λ offspring from μ parents
        • Evaluate the λ offspring
        • Replace the population with μ individuals from parents and offspring
3. Output Best individual or population found.
```

2.3.3 Common Concepts for Evolutionary Algorithms

The main search components for designing an evolutionary algorithm are discussed next.

Selection Methods

The selection mechanism is one of the main search components in EAs that samples the population to determine which individuals are chosen for mating (reproduction) and how many offspring each selected individual produces. In EAs, fitness assignment to individuals may take two different ways:

• *Proportional fitness assignment* in which the absolute fitness are associated with individuals.

• *Rank-based fitness assignment* in which relative fitness are associated with individuals. For instance, a rank in the population is associated with each individual according to its rank in a decreasing sorting of individuals.

The parents are then selected according to their fitness by means of one of the following strategies: roulette wheel selection, stochastic universal sampling, tournament selection and rank-based selection.

Reproduction

Once the selection of individuals to form the parents is performed, the role of the reproduction phase is the application of variation operators such as the mutation and crossover operators.

Mutation

Mutation operators are unary operators acting on a single individual. Mutations represent small changes of selected individuals of the population. The probability p_m defines the probability to mutate each element (gene) of the representation. In general, small values are recommended for this probability ($p_m \in [0.001, 0.01]$). Some strategies initialize the mutation probability to 1/k where k is the number of decision variables, that is, in average only one variable is mutated. Some important points that must be taken into account in the design or use of a mutation operator are as follows:

Ergodicity: The mutation operator should allow every solution of the search space to be reached.

Validity: The mutation operator should produce valid solutions. This is not always possible for constrained optimization problems.

Locality: The mutation should produce a minimal change. The size of mutation is important and should be controllable. The main property that must characterize a mutation operator is *locality*. Locality is the effect on the solution (phenotype) when performing the move (perturbation) in the representation (genotype). When small changes are made in the genotype, the phenotype must reveal small changes. In this case, the mutation is said to have a strong locality. Hence, an evolutionary algorithm will carry out a meaningful search in the landscape of the problem. Weak locality is characterized by a large effect on the phenotype when a small change is made in the genotype. In the extreme case, the search will converge toward a random search in the landscape.

For real-valued vectors, there are many distinct mutation operators. The most used class of mutation operators has the form:

$$x' = x + M$$

In the above equation, *M* is a random variable and x is a candidate solution undergoing mutation. The value of M can take the following different forms:

Uniform Random Mutation

A uniform random variable in the interval [a, b] is generated. The parameter a is in general equal to −b. The offspring is generated within the hyper box x + U(−b, b), where b represents a user-defined constant.

Normally Distributed Mutation

A Gaussian distribution M = N(0, σ) is used, where N(0, σ) is a vector of independent random Gaussian numbers with a mean of 0 and standard deviation σ. It is the most popular mutation scheme in evolution strategies and real-coded genetic algorithms [Tal09].

Other mutation operators such as Cauchy distribution and Laplace distribution are also used at times. The main question here is the initialization of the step size: static or adaptive. In static step size, the algorithm uses the same value during the search, while in adaptive step size; the values are dynamically updated according to the search memory.

Self-adaptive Mutation in Evolution Strategies

In continuous optimization problems, no single step size can efficiently search all dimensions. The mutation scheme for continuous optimization should dynamically scale the mutation strength (step width) to suit each variable. In evolution strategies, the answer provided is the use of self-adaptation to scale and orient the mutation vectors. Each solution vector is paired with a strategy vector that is used to scale the variation operation.

The CMA Evolution Strategy

CMA-ES is one of the most successful optimization algorithms to solve continuous optimization problems. In the CMA-ES, individual step sizes for each coordinate or correlations between coordinates are governed by covariance matrices [Tal09]. CMA-ES adapts the covariance matrix of the multivariate normal mutation distribution. The mutation distribution conducts the generation of new candidate solutions. CMA-ES is a second-order optimization approach, that is, it captures dependencies between variables. The covariance matrix defines the pairwise dependencies between the variables in the distribution. Adaptation of the covariance matrix is based on learning a second-order model of the target objective function, which is reduced to the approximation of the inverse Hessian matrix in the quasi-Newton method, a traditional method in continuous optimization. The CMA-ES is based on two adaptation principles:

Maximum Likelihood: The idea behind this principle is to increase the probability of a successful mutation step. For this purpose, the algorithm updates the covariance matrix of the distribution such that the likelihood of already applied successful steps is increased. Then, the CMA-ES algorithm uses an iterated PCA (principal components analysis) of successful mutation steps while retaining all principal axes.

Evolution Path: The other adaptation principle is based on memorizing the time evolution path of the distribution mean. This path will contain important information of the correlation between successive steps. During the search, the evolution path is used for the covariance matrix adaptation procedure in place of single successful mutation steps. Moreover, the evolution path is used to apply an additional step-size control. The goal of this step-size control is to make successive moves of the distribution mean orthogonal in expectation.

Recombination or Crossover

Unlike unary operators such as mutation, the crossover operator is binary and sometimes *n*-ary. The role of crossover operators is to inherit some characteristics of the two parents to generate the off springs. As for the mutation operator, the design of crossover operators mainly depends on the representation used. The main characteristic of the crossover operator is *heritability*. The crossover operator should inherit a genetic material from both parents. Real-recombination operators use probability distributions around the parent solutions to create offspring. Some operators emphasize solutions at the centre of mass of parents and some emphasize solutions around the parents. Among numerous studies on development of different recombination operators, *blend crossover* (BLX), *simulated binary crossover* (SBX), *uni-modal normal distribution crossover* (UNDX) and *simplex crossover* (SPX) are commonly used. A number of other recombination operators such as arithmetic crossover are also commonly used. A detailed study of many such operators was carried out in [HLM98]. In the recent past, GAs with some of these recombination operators have been demonstrated to exhibit self-adaptive behaviour similar to that in evolution strategy and evolutionary programming approaches [DJA02].

Beyer et al., in [BD01] argued that a recombination operator should have the following two properties:

1. Population mean decision vector should remain the same before and after the recombination operator.

2. Variance of the intra-member distances must increase due to the application of the recombination operator.

The population-mean-decision vector should remain same before and after the recombination since the recombination operator does not use any fitness function information explicitly. Secondly, the selection operator has a tendency to reduce the population variance, thus the population variance must be increased by the recombination operator to preserve adequate diversity in the population.

In the mean centric recombination approach, the population mean is preserved by having individual recombination events preserving the mean between the participating parents and resulting offspring. In the parent centric recombination approach the offspring are created near the parents, however each parent is assigned an equal probability of creating offspring in its neighbourhood. This ensures that the population mean of the entire offspring population is identical to that of the parent population.

Recombination operators such as *uni-modal normal distribution crossover* (UNDX), *simplex crossover* (SPX), and *blend crossover* (BLX) are mean-centric approaches, whereas the *simulated binary crossover* (SBX) and fuzzy recombination in [VMC95] are parent-centric approaches.

In [DJA02], Deb et al., carried out a performance comparison between different recombination operators and showed the superiority of the UNDX and the *multi-parent centric* recombination (mPCX) over other recombination operators. In this thesis, these recombination operators will be used. We have also included the arithmetic recombination due to its relative ease of implementation to initially test our hypothesis, and the simple n-point recombination is used because of its similarity to the binary crossovers. A brief description of these operators follows.

Arithmetic Recombination

The arithmetic recombination operator attempts to average the elements of the parent. The selection operator samples parents for recombination and the arithmetic recombination then creates an offspring based on some parameter α. Given two parents $x_1$and x_2, the arithmetic recombination operator create an off spring using the weighted average:

$$o_i = \alpha x_{1i} + (1-\alpha)x_{2i} \tag{2.17}$$

Uni-modal Normal Distribution Crossover (UNDX)

The *uni-modal normal distribution crossover* (UNDX) operator proposed by Ono et al., in [OK97] uses multiple parents and creates offspring solutions around the centre of mass of these parents. A small probability is assigned to solutions away from the centre of mass.

In this mean-centric crossover operator, (μ - 1) parents x^i are randomly selected from the population. The mean value 'g' of the selected individuals is this computed. Then, (μ - 1) direction vectors, $d^i = x^i - g$ are generated. The variable e^i, denotes direction cosines $d^i / \left| d^i \right|$. Given a randomly selected individual x^n, the length D of the vector $(x^{\mu} - g)$ orthogonal to all $e^i$is calculated.

Let e^i(j = μ... n) be the orthonormal basis of the subspace orthogonal to the subspace spanned by all e^i $(i = 1, \dots, \mu - 1)$ where n represents the size of the individuals. The offspring is generated as follows:

$$y = g + \sum_{i=1}^{\mu-1} w_i \left| d^i \right| e^i + \sum_{i=\mu}^{n} v_i D e^i \tag{2.18}$$

Here w_i and v_i are standard zero-mean normally distributed variables. The UNDX operator will be discussed in detail in chapter 7.

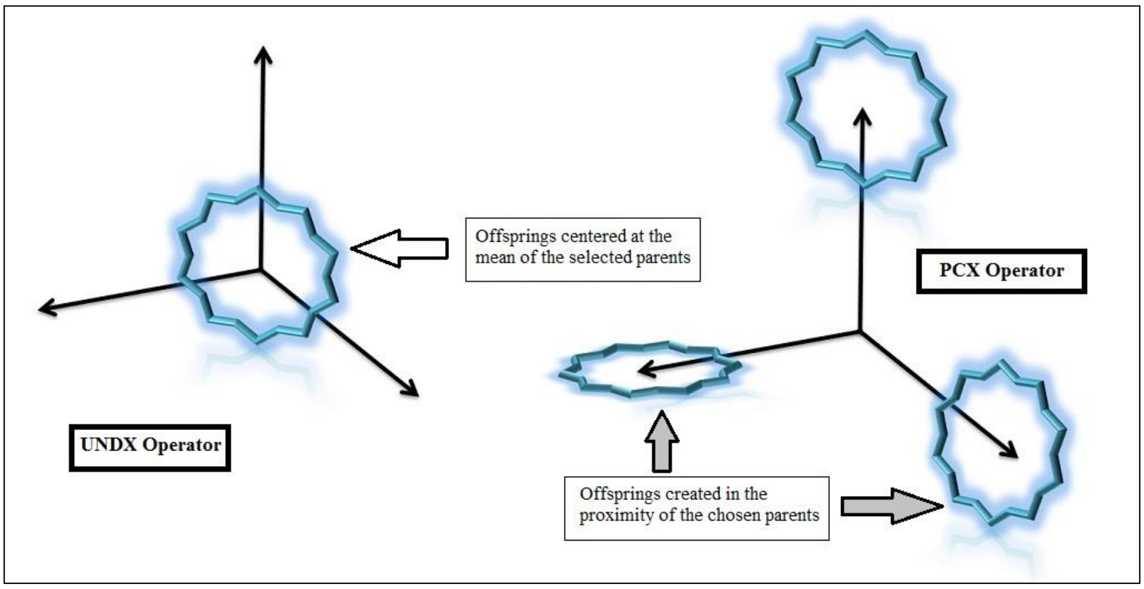

Offsprings centered at the mean of the selected parents
PCX Operator
UNDX Operator
Offsprings created in the proximity of the chosen parents

Figure 2-12: Mean-Centric and Parent-Centric Recombination Operators

Parent Centric Crossover (PCX)

In [DJA02], Deb et al., proposed the *parent-centric recombination* operator (PCX) and compared its performance with a couple of commonly used mean-centric recombination operators (UNDX and SPX). Using steady-state, elite-preserving, and computationally fast models, they showed the superiority of PCX over mean-centric operators. This operator creates the offspring by following these steps:

- 1. First the mean vector g is calculated.
- 2. Then one parent $\boldsymbol{x^p}$is selected in equal probability for each off spring.
- 3. The direction vector $\boldsymbol{d^p} = \boldsymbol{x^p} - \boldsymbol{g}$ is then computed.
- 4. From each of the other μ -1 parents, perpendicular distances $\boldsymbol{D_i}$ to the line $\boldsymbol{d_p}$ are computed and their average D is found.
- 5. The off spring is generated as follows:

$$y = x^p + w_E|d^p| + \sum_{i=1,i \neq p}^{\mu} w_n D e^i \tag{2.19}$$

Here e^irepresents the *(μ - 1)* orthonormal basis spanning the subspace perpendicular to d^p.

Simple N-Point Recombination

In one-point recombination, a single crossover point on both parents' organism strings is selected. All data beyond that point in either organism string is swapped between the two parent organisms. Similarly for n-point recombination, n crossover points on both parent strings are selected. This recombination operator is widely used in binary-coded GAs, and the initial analysis performed by Holland when formulating the schema theorem was based on this operator.

Although N-point recombination operator is not recommended for real-optimization problems [Tal09], it has been included here since it is easier to visualize and makes it easier to convey the key concepts of the building-block hypothesis to the reader.

The Simple N-point recombination is shown in the figure 2-13.

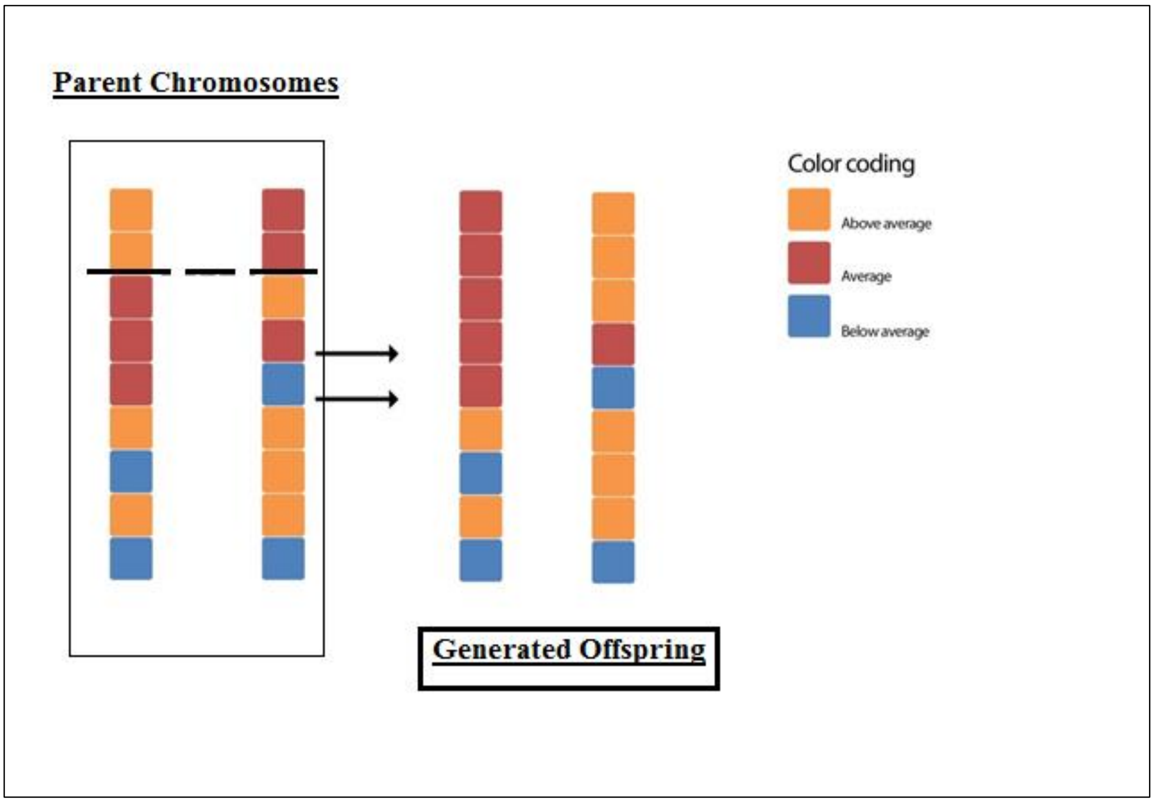


Figure 2-13: A Simple N-Point Recombination

According to Deb et al., in [DJA03] the parent-centric recombination is the most efficient real recombination operator. However it remains to be seen how these operators will compare in performance within a particle filtering scenario. This comparison is carried out in chapter 6 of this thesis.

Replacement Strategies

The replacement phase concerns the survivor selection of both the parent and the offspring populations. As the size of the population is constant, it allows to withdraw individuals according to a given selection strategy.

Generational Replacement: This replacement will concern the whole population of size μ. The offspring population will replace systematically the parent population. This strategy is applied in the canonical GA as proposed by Holland.

Steady-State Replacement: At each generation of an EA, only one offspring is generated. For instance, it replaces the worst individual of the parent population. Between those two extreme replacement strategies, many distinct schemes that consist in replacing a given number of λ individuals of the population may be applied ($1 < \lambda < \mu$). Elitism always consists in selecting the best individuals from the parents and the off springs. This approach leads to a faster convergence and a premature convergence could occur. Sometimes, selecting bad individuals is necessary to avoid the sampling errors. Those replacement strategies may be stochastic or deterministic.

Although GAs are simple to describe and program, their behaviour can be complicated, and many open questions exist about how they work and for what types of problems they are best suited [Mit98]. The next subsection introduces the Schema theorem that was formulated by Holland to explain the working of a GA.

2.3.4 The Schema Theorem

The traditional theory of GAs, first formulated in [Hol75], assumes that, at a very general level of description, GAs work by discovering, emphasizing, and recombining good "building-blocks" of solutions in a highly parallel fashion. The idea here is that good solutions tend to be made up of good building-blocks; combinations of alleles that confer higher-fitness on the strings in which they are present. Holland introduced the notion of *schemas* (or *schemata*) to formalize the informal notion of building-blocks.

In [ES07], Eiben & Smith noted that Holland used an aggregation approach to model the working of a GA. They noted:

*a schema, in a binary setting, as a set of bit strings that can be described by a template made up of ones, zeros, and asterisks, the asterisks representing 'don't cares'. For example, the schema H = 1 * * * * 1 represents the set of all 6-bit strings that begin and end with 1. Holland's initial work showed that the analysis of GA behaviour was far simpler if carried out in terms of schemata [Hol75]. This is an example of aggregation in which rather than model the evolution of all possible strings, they are grouped together in a way that the evolution of the aggregated variables is modelled [ES07].*

Holland showed that a string of length l is an example of 2^l schemata. In a population of μ individuals the population will usefully process $O(\mu^3)$ schemata. This result is known as implicit-parallelism and is quoted as one of the main reasons of the success of genetic algorithms [ES07].

Consider Holland's analysis applied to the *standard genetic algorithm* (SGA) using fitness-proportionate parent selection, one-point crossover (1X), and bitwise-mutation, with a generational schema for survivor selection. A genotype of length *l* that contains an example of a schemata *H*, the schema may be disrupted if the crossover point falls between the ends, which happens with probability:

$$Pd(H, 1X) = \frac{d(H)}{(l-1)} \tag{2.20}$$

The probability that bitwise mutation with probability P_m will disrupt the schema H is proportional to the order of the schema, *O(H)*: $Pd(H, mutation) = 1 - (1 - P_m)^{o(H)}$, which after expansion and ignoring high-order terms in P_m approximates to:

$$Pd(H, mutation) = o(H).P_m \tag{2.21}$$

The probability of a schema being selected depends on the fitness of the individuals in which it appears relative to the total population fitness, and the number of examples present *n(H, t)*. Using *f(H)* to represent the fitness of the schema *H*. < *f* > denotes the mean population fitness.

$$P_s(H,t) = \frac{n(H,t).f(H)}{\mu.< f >} \tag{2.22}$$

Noting that μ independent samples are taken to create the next set of parents, the expected number of instances of *H* in the population after selection is then:

$$n'(H,t) = \mu.P_s(H,t) = \frac{n(H,t).f(H)}{< f >} \tag{2.23}$$

After normalizing by μ, to make the result population size independent, allowing for the disruptive effects of recombination and mutation derived above, and using an inequality to allow for the creation of new instances of *H* by the variation operators, the proportion *m(H)* of individuals representing schema at subsequent time steps is given by:

$$m(H,t+1) \geq m(H,t)\frac{f(H)}{< f >}.\left[1 - \left(P_c.\frac{h(H)}{l-1}\right)\right].[1 - p_m.o(H)] \tag{2.24}$$

The equation 4.5 is the well-known Schema theorem for binary-encoded GAs. The equation shows that above-average fitness schemas increase in number as the algorithm proceeds to run.

2.3.5 Schemas and the Two Armed Bandit Problem

Holland's original motivation for developing GAs was to construct a theoretical framework for adaptation as seen in nature, and to apply it to the design of artificial adaptive systems. According to Holland, an adaptive system must persistently identify, test and incorporate structural properties hypothesized to give better performance in some environment. Schemas are meant to be a formalization of such structural properties. In the context of genetics, schemas correspond to constellations of genes that work together to effect some adaptation in an organism; evolution discovers and propagates such constellations [Mit98].

Implicit Parallelism

Holland's schema analysis showed that a GA, while explicitly calculating the fitness of the μ members of a population, implicitly estimates the average fitness of a much larger number of schemas by implicitly calculating the observed average fitness of schemas with instances in the population. It does this without needing any additional memory or computation time beyond that needed to process the μ members of the population. Holland called this implicit-parallelism.

Holland's analysis also showed that those schemas whose fitness estimates remain above average receive increasing numbers of instances in the population; the Schema theorem has been interpreted to imply that, under a GA short low-order schemas whose average fitness remains above the mean will receive exponentially increasing numbers of samples over time.

Building Block Hypothesis

Holland's analysis suggests that selection increasingly focuses the search on subsets of the search space with estimated above-average fitness, whereas recombination puts high-fitness building-blocks together on the same string in order to create strings of increasingly higher fitness. This is called the building-block hypothesis. Mutation plays the role of an insurance policy, making sure genetic diversity is never irrevocably lost at any locus [Mit98].

Holland framed adaptation as a tension between exploration and exploitation [Mit98]. The tension comes about since any move toward exploration, testing previously unseen schemas or schemas whose instances seen so far have low fitness, takes away from the exploitation of tried and true schemas. In a system required to face environments with some degree of unpredictability, an optimal balance between exploration and exploitation must be found. The system has to keep trying out new possibilities but it also has to continually incorporate and use past experience as a guide for future behaviour. Holland used a Two-Armed Bandit analogy to describe this phenomenon in GAs.

Two-Armed Bandit Problem

Holland's schema analysis demonstrated that, given certain assumptions, the GA indeed achieves a near-optimal balance. Holland's arguments for this are based on an analogy with the Two-Armed Bandit problem.

The trade-off between exploration and exploitation can be instructively modelled in a simple scenario: the Two-Armed Bandit problem. This problem has been studied extensively in the context of statistical decision theory and adaptive control [Bel61]. Holland used it as a mathematical model of how a GA allocates samples to schemas. The scenario is as follows. A gambler is given μ coins with which to play a slot machine having two arms. The arms are labelled A_1 and A_2, and they have mean payoff (per trial) rates P1 and P2 with respective variances Ã$_{11}$ and Ã$_{22}$. The payoff processes from the two arms are each stationary and independent of one another, which means that the mean payoff rates do not change over time. The gambler does not know these payoff rates or their variances; but can estimate them only by playing coins on the different arms and observing the payoff obtained on each. The gambler has no *a priori* information on which arm is likely to be better. The goal is to maximize the total payoff during the μ trials.

Note that the goal is not merely to guess which arm has a higher payoff rate, but to maximize payoff in the course of gaining information through allocating samples to the two arms. Such a performance criterion is called on-line, since the payoff at every trial counts in the final evaluation of performance. This is to be contrasted with the common off-line performance criteria in function optimization, where the performance evaluation of an optimization method might depend only on whether or not the global optimum was discovered, or possibly on the best fitness level achieved after a given number of trials, irrespective of the fitness (payoff) of the intermediate samples.

Holland's analytic solution to the Two-Armed Bandit problem states that, as more and more information is gained through sampling, the optimal strategy is to exponentially increase the probability of sampling the better-seeming arm relative to the probability of sampling the worse seeming arm. To apply this to schema sampling in a GA, the 3^L schemas in an *L*-bit search space can be viewed as the 3^L arms of a multi-armed slot machine. The observed payoff of a schema *H* is simply its observed average fitness, which the GA implicitly keeps track of via the number of samples of *H* in the population. Holland's claim, supported by the Schema Theorem, is that, under the GA a near-optimal strategy for sampling schemas arises implicitly, which leads to the maximization of on-line performance.

The Two-Armed Bandit problem is a simple model of the general problem of how to allocate resources in the face of uncertainty. This is the *exploration versus exploitation* problem faced by an adaptive system. The Schema Theorem suggests that, given a number of assumptions, the GA roughly adopts a version of the optimal strategy described above over time, the number of trials allocated to the best observed schemas in the population increases exponentially with respect to the number of trials allocated to worse observed schemas. The GA implements this search strategy via implicit-parallelism, where each of the *n* individuals in population can be viewed as a sample of 2^l different schemas. The number of instances of a given schema *H* in the population at any time is related to its observed average performance, giving an exponential growth rate for highly fit schemas.

This leads to the following qualitative formulation of the Schema Theorem and the Building Block Hypothesis taken from [Mit98]:

"The simple GA increases the number of instances of low-order; short-defining length, high−observed−fitness schemas via the multi−armed−bandit strategy, and these schemas serve as building-blocks that are combined, via crossover, into candidate solutions with increasingly higher order and higher observed fitness."

2.3.6 Constructive Ability of Recombination Operators

The Schema Theorem, by itself, addresses the positive effects of selection but only the negative aspects of recombination and mutation i.e., the extent to which they disrupt schemas. It does not address the question of how recombination works to recombine highly-fit schemas, even though this is the major source of the search power of genetic algorithms. The Building-Block Hypothesis states that recombination combines short, observed high-performance schemas into increasingly fit candidate solutions, but does not give any detailed description of how this combination takes place. To investigate schema processing and recombination in more detail, in [MFH92, MHF94], Mitchel et al., designed a class of fitness-landscapes, called Royal Road functions, that were meant to capture the essence of building-blocks in an idealized form.

The Building-Block Hypothesis suggests two features of fitness landscapes that are particularly relevant to genetic algorithms: the presence of short, low-order, highly-fit schemas; and the presence of intermediate stepping stones i.e., intermediate-order higher-fitness schemas that result from combinations of the lower-order schemas and that, in turn, can combine to create even higher-fitness schemas. In [MFH92] the authors carried out an experiment to validate the building-block hypothesis. The authors compared a GA performance with a random-mutation hill-climbing algorithm [RMHC]. However the results of their experiment showed the RMHC to outperform the GA. In their experiment the RMHC took on average 6179 iterations to converge while the GA took 61,334 iterations.

One reason for the poor performance of the GA was *hitch-hiking*; once an instance of a higher-order schema is discovered, its high fitness allows the schema to spread quickly in the population, with zeros in other positions in the string hitch-hiking along with the ones in the schema's defined positions. This slows the discovery of schemas in the other positions, especially those that are close to the highly fit schema's defined positions. In short, hitch-hiking seriously limits the implicit-parallelism of the GA by restricting the schemas sampled at certain loci. To overcome this issue, the authors proposed the following considerations to be made to a genetic algorithm:

Independent Samples

The population has to be large enough, the selection process has to be slow enough, and the mutation rate has to be sufficiently high to make sure that no single locus is fixed at a single value in every string in the population, or even in a large majority of strings.

Sequestering Desired Schemas

Selection has to be strong enough to preserve desired schemas that have been discovered, but it also has to be slow enough (or, equivalently, the relative fitness of the non over lapping desirable schemas has to be small enough) to prevent significant hitchhiking on some highly fit schemas, which can crowd out desired schemas in other parts of the string.

Instantaneous Recombination

The recombination rate has to be such that the time for a crossover that combines two desired schemas to occur is small with respect to the discovery time for the desired schemas.

These mechanisms were not all mutually compatible (e.g., high mutation works against sequestering schemas), and thus they must be carefully balanced against one another. These balances are discussed in [Hol93], and work on using such analyses to improve the GA was reported in [MHF94]. With these considerations, the authors carried out the same experiment and by comparison, the GA was able to converge in only 696 iterations. According to the authors, the result of this experiment increased the plausibility of the building-block like effects being responsible for the better performance of the GA.

Similarly Spears in his PhD thesis, [Spe98], analysed the constructive ability of recombination and mutation. He showed that with everything else constant, the construction of high-order hyperplanes increased as the recombination rate $P_o$was increased, with maximum construction occurring at $P_o = 0.5$. He also showed that for a mutation operator, construction of high-order hyperplanes decreases with increasing mutation rate, with maximum construction occurring when mutation rate is equal to 0.

2.3.7 Real-Coded Genetic Algorithms

Real-coded genetic algorithms (rGAs) use real numbers to represent the genes. Encoding is a key issue in GA work because GAs directly manipulates a coded representation of the problem and because the encoded schema can severely limit the window by which a system observes its world [Koz92]. Fixed length and binary coded strings for the representation solution have dominated GA research since there are theoretical results that show them to be the most effective ones [Gol91], and as they are amenable to simple implementation. For optimization in the continuous domain, it would seem particularly natural to represent the genes directly as real numbers. Then a chromosome is a vector of floating point numbers. The size of the chromosomes is kept the same as the length of the vector which is the solution to the problem; in this way each gene represents a variable of the problem. The values of the genes are forced to remain in the interval established by the variables which they represent, so the genetic operators must observe this requirement. Enhanced schema processing is obtained by using alphabets of low-cardinality; however this is a direct contradiction of the results obtained when rGAs were applied in many real world applications [Gol89, Gol91, HH98].

Real-coded genetic algorithms have been successfully used in a wide variety of applications in business, engineering and science [HH98, Gol89, Gol91]. The behaviour of rGAs depends to a large extent on many factors such as population size, genetic operators and the values of their parameters, to name a few. In this regard, several investigators have focused on the theoretical underpinnings of rGAs.

2.3.8 The Theory of Virtual Alphabets

In [Gol91], Goldberg postulated his theory to explain the workings of rGAs. Goldberg investigated the convergence property of rGAs through the concept of a virtual alphabet. The theory suggested how the process of selection quickly reduces the cardinality of actual alphabets that are discovered by recombination. It also suggested that rGAs may be blocked from further progress when local optima decouple the virtual characters from the global optimum.

Goldberg showed that his theory is consistent with the theory of schemata and postulated that selection dominates early GA performance and restricts subsequent search to intervals with above-average function value dimension by dimension. These intervals may be further subdivided on the basis of their attraction under genetic hill climbing. Each of these subintervals is called a virtual character and the collection of characters along a given dimension is called a virtual alphabet. It is the virtual alphabet that is searched during the recombinative phase of the GA; these alphabets are combined via the building-block hypothesis, similar to binary-coded GAs.

Virtual characters and alphabets provide a useful perspective from which to view the convergence mechanisms of real-coded GAs. Simply restated, one-dimensional basin features are selected early in the GA dimension-by-dimension, and the collection of virtual alphabets thus selected is used in subsequent recombinative-selective search. This mechanism seems to side step the precision and aliasing problems that may occur when low-cardinality codes are used by allowing real GAs to adaptively select their own alphabets. The empirical success enjoyed by users of Evolutions strategies and real-coded genetic algorithms can in large part be explained by this single factor.

In the next section, the *Stochastic Volatility* (SV) estimation problem is discussed. This estimation problem is used as a test problem in this thesis. The algorithms present in literature for this particular problem are then discussed in the concluding section of this chapter. These algorithms will be used as benchmark for comparison in this thesis.

2.4 The Stochastic Volatility Estimation Problem

The estimation of the volatility of common stocks is used as a test problem in this thesis to compare the performance of the particle filtering algorithms under changing dimensions and particle population size. This section lays the foundation of the stochastic volatility estimation problem and discusses the two benchmark filtering algorithms that will be used in this thesis for comparison.

2.4.1 Option Pricing & Stochastic Volatility Models

Options are financial contracts where the price of the contract is based on the price of an underlying asset. The underlying asset is assumed to follow a Brownian process. The price of the option contract is equal to the pay-off of the contract, i.e., the difference in the strike price and the actual price of the underlying asset at maturity [Jo08].

Options were designed to hedge against risk in the financial markets. The holder of the option locks in on the price of the underlying asset, and in case the price fluctuates and moves in the direction that is not favourable, the holder exercises the option and executes the trade on the price agreed in the option contract. The price of an option is equivalent to:

$$P_t \propto E(S_T) - K \tag{2.25}$$

Here P_t is the price of the option contract, $E(S_T)$ is the expected value of the underlying asset at the expiry date of the option T and K is the strike price, i.e., the price agreed at which the trade will take place.

Much of the research in financial literature is based on accurately modelling the behaviour of the underlying stochastic process followed by S_t. Most of the models proposed to model the behaviour of assets assume that the asset follows a Brownian motion with a time dependant drift. Mathematically:

$$\frac{dS}{S} = \mu dt + V^{1/2} dW \tag{2.26}$$

Here V is the volatility of the asset and W is Brownian random noise.

Estimating the Stochastic Volatility of Assets

The assumption of a constant volatility in equation 2.26 has drawn much criticism, since the instantaneous volatility of a stock is itself a stochastic quantity. Thus during certain periods, more information arrives causing the stock to wobble rapidly. During such a period the total amount of fluctuation expected during the option's life will be greater, and one therefore expects the option's cost to be higher [JPR95]. Hence it was proposed that this model should be extended and the volatility should also be stochastic [Hu01]. Thus the option pricing formula was redeveloped keeping the volatility stochastic.

Volatility is generally chosen to follow a diffusive process though in more sophisticated models, it can be allowed to jump, and indeed some models mix jump-diffusion and *stochastic volatility* (SV) to reflect the greater volatility in the market. The pricing model consists of a coupled differential equation:

$$\frac{dS}{S} = \mu dt + V^{1/2} dW^{(1)} \tag{2.27}$$

$$dV = \mu_V dt + \sigma_V V^{\alpha} W^{(2)} \tag{2.28}$$

Here α is a positive constant, and $W^{(1)}$ and $W^{(2)}$ can be correlated or uncorrelated Brownian motions. The volatility obtained by the above equation is then used to price the option [Jo08, Hu01].

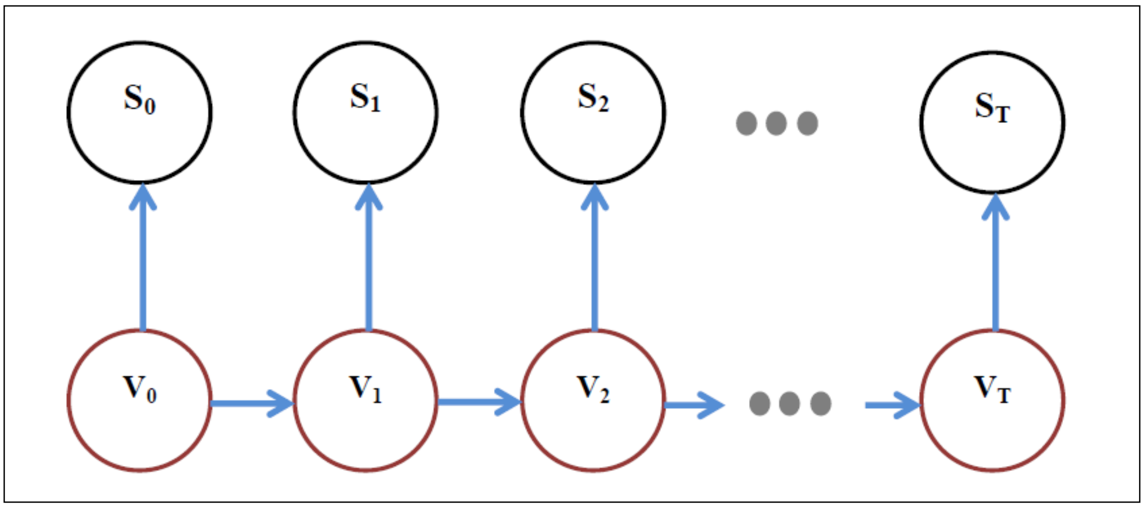


Figure 2-14: The Stochastic Volatility State Space

2.4.2 Calibration of the Stochastic Volatility Models

Stochastic volatility models are calibrated to market prices. Monte Carlo algorithms have provided a flexible and powerful tool for the inference on complex models, possibly with non-observable components [J95PR]. The use of *Markov Chain Monte Carlo* (MCMC) methods for the calibration of stochastic volatility models started with the important paper by Jacquier, Polson and Rossi in [JPR95]. In this paper they based their algorithm on the basic log-stochastic volatility model, and used the return price process for calibration. Their model was based on estimating the stationary distribution of the parameters in the volatility equation. Once the distribution of the parameters converges to a stationary distribution, an approximation of the volatility is also obtained.

Although MCMC methods provide an accurate and efficient estimate of the state and parameters, they are inefficient when dealing with online price updates [LW01]. The constant arrival of prices for all assets during trading times requires an online algorithm that should be able to update the parameters and estimate the states. Due to the non-linear nature of the state-space, particle filters are used for the online approximation, however their performance is affected with the increase in the model parameters [LW01]. Increasing model parameters contributes to an increase in the state dimension, and this leads to sample degeneracy in the particle filter. For this very reason, the prediction cannot be parallelised by running a particle filter on multiple time series.

2.4.3 Particle Filters for SV Estimation – The Bench Mark Algorithms

Two particle filtering algorithms are discussed in this section. These algorithms will be used as benchmarks in this thesis.

The Particle Filter of Liu and West

An approach for the filtering problem of a dynamic state space model based on the concept of artificial evolution [GSS93] has been proposed by Liu et al., in [LW01]. Given a parameter θ and observation vector *D*, their approach generalizes in a dynamic context the kernel smoothing approximation of the posterior $p(\theta|D_t)$. According to artificial evolution, at time t+1, after the resampling step an independent zero-mean normal increment is added to the parameter.
That is:

$$\theta_{t+1} = \theta_t + \varepsilon_{t+1} \tag{2.29}$$

$$\varepsilon_{t+1} \sim N(\theta, \sigma)$$

The key motivating idea is that the artificial evolution provides the mechanism for generating new parameter values at each time step. The undesirable loss of information implicit in equation (2.29) can be easily quantified. The Monte Carlo approximation to $p(\theta|D_t)$ has mean θ_t and variance matrix V_t. Hence, in the evolution in equation (2.29), the implied prior $p(\theta_{t+1}|D_t)$ has the correct mean but variance matrix $V_t + W_{t+1}$.The loss of information is explicitly represented by the component $W_{t+1.}$

Liu et al., addressed the loss of information that results because of the addition of noise. Assuming a non-zero covariance matrix, the artificial evolution equation implies:

$$V(\theta_{t+1}|D_t) = (\theta_t|D_t) + W_{t+1} + 2C(\theta_t.\varepsilon_t|D_t) \tag{2.30}$$

The ‘no information’ loss implies:

$$V(\theta_{t+1}|D_t) = V(\theta_t|D_t) = V_t \tag{2.31}$$

This then implies:

$$C(\theta_t.\varepsilon_t|D_t) = -W_{t+1}/2 \tag{2.32}$$

Hence, there must be a structure of negative correlations to remove the unwanted information loss effect. In the case of approximate joint normality of $(\theta_t.\varepsilon_t|D_t)$, this would then imply the conditional normal evolution in which :

$$p(\theta_{t+1}|\theta_t) = N\left(\theta_{t+1}\middle|A_{t+1}\theta_t + (I - A_{t+1})\hat{\theta}_t, (I - A_{t+1}^2)V_t\right)$$

Here

$$A_{t+1} = I - W_{t+1}V_t^{-1}/2 \tag{2.32}$$

The generalized Monte Carlo approximation to $p(\theta_{t+1}|D_t)$ is then a generalized kernel form with complicated shrinkage patterns induced by the shrinkage matrix A_{t+1}. Liu et. Al., proposed an approximation to the above parameter evolution equation:

$$p(\theta_{t+1}|\theta_t) \sim N(\theta_{t+1}|a\theta_t + (I - a)\bar{\theta}_t, h^2V_t) \tag{2.33}$$

Here

$$h^2 = 1 - a^2 \tag{2.34}$$

So that

$$h^2 = 1 - ((3\delta - 1)/2\delta)^2 \tag{2.35}$$

Also note that $a = \sqrt{1 - h^2}$.

The resulting Monte Carlo approximation to $p(\theta_{t+1}|D_t)$ is then precisely of kernel form with a discounting smoothing factor. A problem encountered with this algorithm is that the estimated variance-covariance matrix $V_t$collapses to zero after a few hundred iterations. In this thesis, the filtering algorithm of Liu et al., will be referred to as PF-LW.

Algorithm 2.5: The SV Estimation Algorithm of Liu and West

```
Choose a proposal distribution $q(x_{k+1}^i | x_k^i, y_{k+1})$, resampling strategy and the number of particles N.
Initialization: Generate $x_1^i \sim p_{x_o}, i = 1, \dots, N$ and let initial weights to be 1/N.

For loop $k$ = 1, 2…End of Observations
Start

    1. For each i = 1, …N, identify the prior point estimates of $(x_t, \theta)$ given by $(\mu_{t+1}^{(j)}, m_t^{(j)})$ where
           $$\mu_{t+1}^{(j)} = E(x_{t+1} | x_t^j, \theta_t^{(j)})$$
       may be computed from the state evolution density and
           $$m_t^{(j)} = a\theta_t^{(j)} + (1-a)\bar{\theta}_t$$
       is the kernel location.
    2. Sample a new parameter vector $\theta_{t+1}^{(k)}$ from the normal component of kernel density, namely
           $$\theta_{t+1}^{(k)} \sim N(\cdot | m_t^{(k)}, h^2 V_t)$$
    3. Sample a value of the current state vector $x_{t+1}^{(k)}$ from the system equation
           $$p(\cdot | x_t^{(k)}, \theta_{t+1}^{(k)}).$$
    4. Evaluate a corresponding weight
           $$w_{t+1}^{(t)} \propto \frac{p(y_{t+1} | x_{t+1}^{(t)}, \theta_{t+1}^{k})}{p(y_{t+1} | \mu_{t+1}^{(k)}, m_t^{(k)})}$$
    5. IF ($y_k$ is not last observation)
       $k = k + 1$
       Go to step 1.
         Else End For loop.
END
```

The Parameter Learning Algorithm (PLA)

In [RB06], Raggi et al., noted that the basic setup of the PF-LW performed poorly in practice providing unstable estimates of the posterior over time. A second problem noticed was that the estimated posterior variance-covariance matrix collapses to zero after few hundreds iterations. This latter problem was attributed to the sample impoverishment phenomenon caused by the resampling step; particles with high probability are selected many times causing a loss of diversity. They noted that the problem becomes severe when the noise of the latent process is small.

Raggi et al., proposed their *particle learning algorithm* (PLA) that builds on the PF-LW in which they made the following changes:

1. They integrated an MCMC step to prevent the algorithm to degenerate after a number of iterations. The use of MCMC together with particle filters was proposed in [GB01] and has been proven to be an effective combination between the computational advantages of sequential algorithms and the statistical efficiency of the MCMC methods. The introduction of the MCMC step proved useful when dealing with long time series, since it reduced the degeneration troubles connected with sequential Monte Carlo methods [RB06].

2. They also recommended that resampling be carried on every iteration. In the generic particle filter, the resampling step is called only when the variance of the particles reaches some threshold.

3. To increase sample variability it was recommended to recur MCMC moves. This wariness reduced the correlation between particles. This idea had been developed in Gilks et al., in [GB01]. All these particles can be rejuvenated or moved according to a Markov transition with the same posterior as invariant distribution. For this reason it was not necessary a burn-in time for the MCMC step.

The PLA is described next. The PLA and the PF-LW will be used as benchmarks in this thesis. Both these algorithms will be implemented and their performance will be compared under changing population size and state-dimensions.

Algorithm 2.6: The Particle Learning Algorithm

```
Choose a proposal distribution $q(x_{k+1}^i|x_k^i, y_{k+1})$,, resampling strategy and the number of particles N.
Initialization: Generate $x_1^i \sim p_{x_o}, i = 1, \dots, N$ and let initial weights to be 1/N.

For loop $k$ = 1, 2…End of Observations
Start

    1. For each i = 1, …N, identify the prior point estimates of $(x_t, \theta)$ given by $(\mu_{t+1}^{(j)}, m_t^{(j)})$ where
       $$\mu_{t+1}^{(j)} = E\left(x_{t+1}|x_t^j, \theta_t^{(j)}\right)$$
       may be computed from the state evolution density and
       $$m_t^{(j)} = a\theta_t^{(j)} + (1-a)\bar{\theta}_t$$
       is the kernel location.
    2. Sample a new parameter vector $\theta_{t+1}^{(k)}$ from the normal component of kernel density, namely
       $$\theta_{t+1}^{(k)} \sim N\left(\cdot\,|m_t^{(k)}, h^2V_t\right)$$
    3. Sample a value of the current state vector $x_{t+1}^{(k)}$ from the system equation
       $$p\left(\cdot\,|x_t^{(k)}, \theta_{t+1}^{(k)}\right).$$
    4. Evaluate a corresponding weight
       $$w_{t+1}^{(t)} \propto \frac{p\left(y_{t+1}|x_{t+1}^{(t)}, \theta_{t+1}^k\right)}{p\left(y_{t+1}|\mu_{t+1}^{(k)}, m_t^{(k)}\right)}$$
    5. Carry out residual resampling.
    6. (Optional) Move the former particles according to MCMC with invariant
       distribution the posterior and update the sufficient statistics according to
       the former MCMC move.
    7. IF ($y_k$ is not last observation)
       $k = k + 1$
       Go to step 1.
         Else End For loop.
END
```

It will be shown experimentally in chapter 5 that although the PLA performed better compared to the PF-LW, however with an increase in the state-dimensions its performance started to deteriorate significantly.

2.5 Summary

This chapter laid the foundation of the Bayesian filtering theory and the issues currently faced in implementing SMC methods in high-dimensional state spaces. Later, metaheuristics were discussed and the common concepts of evolutionary algorithms were discussed in detail followed by important GA theoretic arguments that we will use in the next chapter.

The last section gave an explanation to the test problem of this thesis, the stochastic volatility estimation of common stocks. In practice after the MCMC calibration of the stochastic volatility estimation model, particle filters are used to estimate the volatility online. However the model parameters also change with time and need to be re-estimated. For this dual-state and parameter estimation problem, the joint distribution of the state and the model parameters needs to be estimated. Two benchmark algorithms, PF-LW and PLA were later discussed that are used commercially for this task. These two algorithms will be used as a benchmark for performance comparison in this thesis. These two filtering algorithms have been shown to perform accurately in low dimensions; however they suffer from sample impoverishment in higher dimensions. In chapter 6 of this thesis, the benchmark algorithms will be implemented and their performance under varying population size and state-dimensions will be compared with our proposed algorithm.

The next chapter lays the foundation of the RGAPF. The approach mentioned in the next chapter follows from the similarities of a particle filter and genetic algorithm. Based on these similarities, genetic algorithm theoretic arguments are used to analyse the reason behind the particle filter collapse and how to address this issue.

Chapter 3

Approach

Bayesian filtering for non-linear and non-Gaussian state-spaces was introduced in chapter 2. As mentioned in the previous chapters, sequential Monte Carlo methods suffer from sample impoverishment as the state-dimensions are increased. Theoretically the number of particles required for correct approximation needs to increase exponentially with the increasing state-dimensions. This requirement makes the practical implementation of particle filters infeasible in most scenarios. The resampling and regularization techniques for sample impoverishment are successful in low-dimensional cases however they are unable to address this issue as the state dimensions are increased [SBBA08, BBL08, Bri11].

In this chapter the similarities between GAs and particle filters are discussed and using GA theoretic arguments it is hypothesized that the addition of a recombination and mutation layer in a particle filter may address the issues faced by particle filters in higher dimensions.

This chapter is divided into the following sections:

- Section 3.1 compares GAs with particle filters and discusses the similarities between them.
- In section 3.2 the phenomenon of sample impoverishment and reasons of collapse of a particle filter in high-dimensions is revisited and discussed using approaches found in GA literature.
- Section 3.3 presents the hypothesis of this thesis and discusses the modifications that need to be made in a generic particle filter that may address the issues it faces in high-dimensions.
- In section 3.4 we propose the real-coded genetic algorithm particle filter (RGAPF).
- In section 3.5 we compare the proposed RGAPF with the generic particle filter.

- The summary of this chapter is given in section 3.6.

3.1 Genetic Algorithms and Particle Filters Compared

In this section, particle filters and genetic algorithms are compared to get a better understanding of the similarities and the difference between the two algorithms. The next table compares both these algorithms.

Table 3-1: Genetic Algorithms & Particle Filters - A comparison

	Particle Filter	Genetic Algorithm	Notes
1	Initialize particle population	Initialize parent population	
2	Assign weights to particles using state update equation & importance sampling.	Assign weights to the population using the fitness function.	For particle filters, this step contains both time and state update.
3	Carry out resampling.	Select parent candidates using a selection operator.	In both the algorithms, candidate particles/parents are selected based on their weight/fitness within the population.
4		Perform recombination	
5	Carry out artificial evolution.	Perform mutation	Mutation in real coded GAs is similar to artificial evolution.
6		Evaluate child chromosome fitness, and insert in the population.	In particle filters the assignment of weights is carried out in step 2.
7	If new observation is received, go to step 2.	Unless stopping criteria has been met, go to step 3	

In the above table, where both the algorithms are compared, all the steps appear similar, however one main step, the recombination, is absent in particle filters. The recombination operator is an important operator in GA literature and as mentioned in the previous section, and in GA literature, the main search and adaptive property of these search algorithms are credited to the recombination operator. To emphasize the importance of recombination operators in a GA, the qualitative formulation of the Schema Theorem and the Building Block hypothesis from [Mit98] is repeated:

*"The simple GA increases the number of instances of low-order; short-defining length, high–observed–fitness schemas via the multi–armed–bandit strategy, and these schemas serve as **building-blocks that are combined, via recombination, into candidate solutions with increasingly higher order and higher observed fitness**."*

It can thus be stated that the building-block hypothesis credits the ability of a recombination operator to work on short-order hyperplanes and combining them to create high-order hyperplanes.

In the next section, the working of a particle filter is analysed from another perspective by considering a particle population as a set of hyperplanes and then analysing the sample impoverishment phenomenon and how it appears in this hyperplane population.

3.2 Filtering in High Dimensions – Constructing Hyperplanes

Consider figure 3-1 which shows the objective of a particle filtering algorithm. In this figure the blue curve represents the posterior to be estimated using SMC methods. A finite set of particles are used to represent samples from this posterior distribution, and the weight assigned to these particles represent the probability of that outcome. The red curve is an approximation of the posterior made using the weights of the particles. The yellow circles in the diagram represent the particles, and the size of the circles give an indication of their weights.

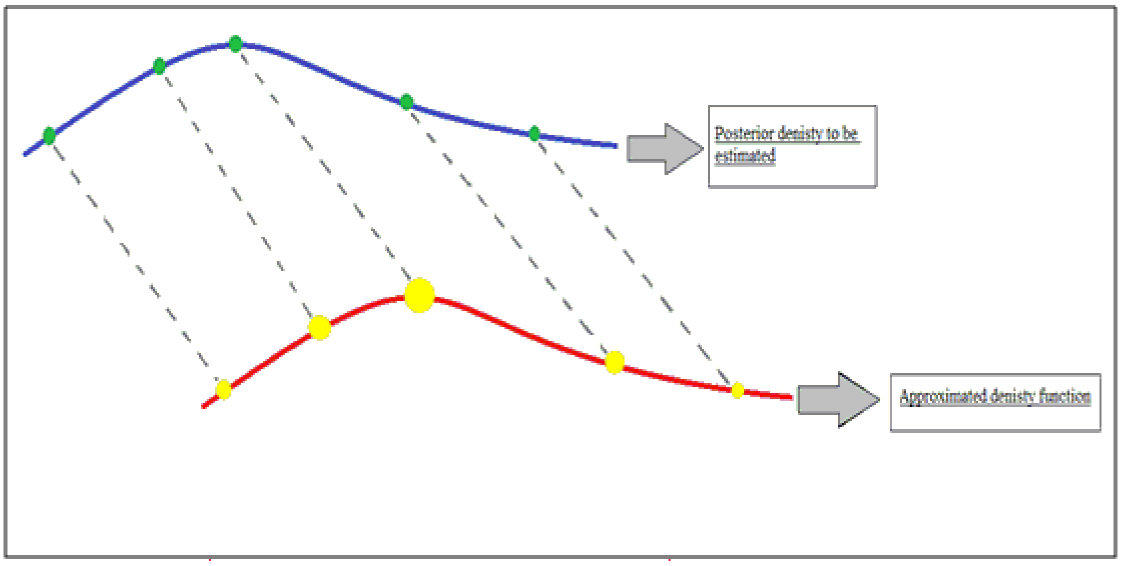

Posterior denisty to be estimated
Approximated denisty function

Figure 3-1: Objective of a Particle Filter

Using a finite set of particle can however result in incorrect posterior density estimation, with the approximated density skewed towards the particle with the highest weight. The size of the particle population and ineffective search operators result in the same particles being used on each iteration, and this results in all but a few particles being assigned negligible weights.

Consider the next diagram. The particles are sampled from the actual posterior, but since the weight assigned is equivalent to the probability of a particular outcome, and since the weights are normalized to add to one, the weight of the particle this represent the relative fitness of the particles within the population, thus the estimated posterior may not be a good approximation to the actual posterior. As the filtering algorithm continues to run, the particles with greater weight tend to get sampled more and hence their weight continues to increase causing the phenomenon of sample impoverishment. Figure 3-2 shows that due to inefficient search, after a few iterations all but one particle kept getting the most number of samples and hence its weight continued to increase, while the weights of other particles continued to decrease.

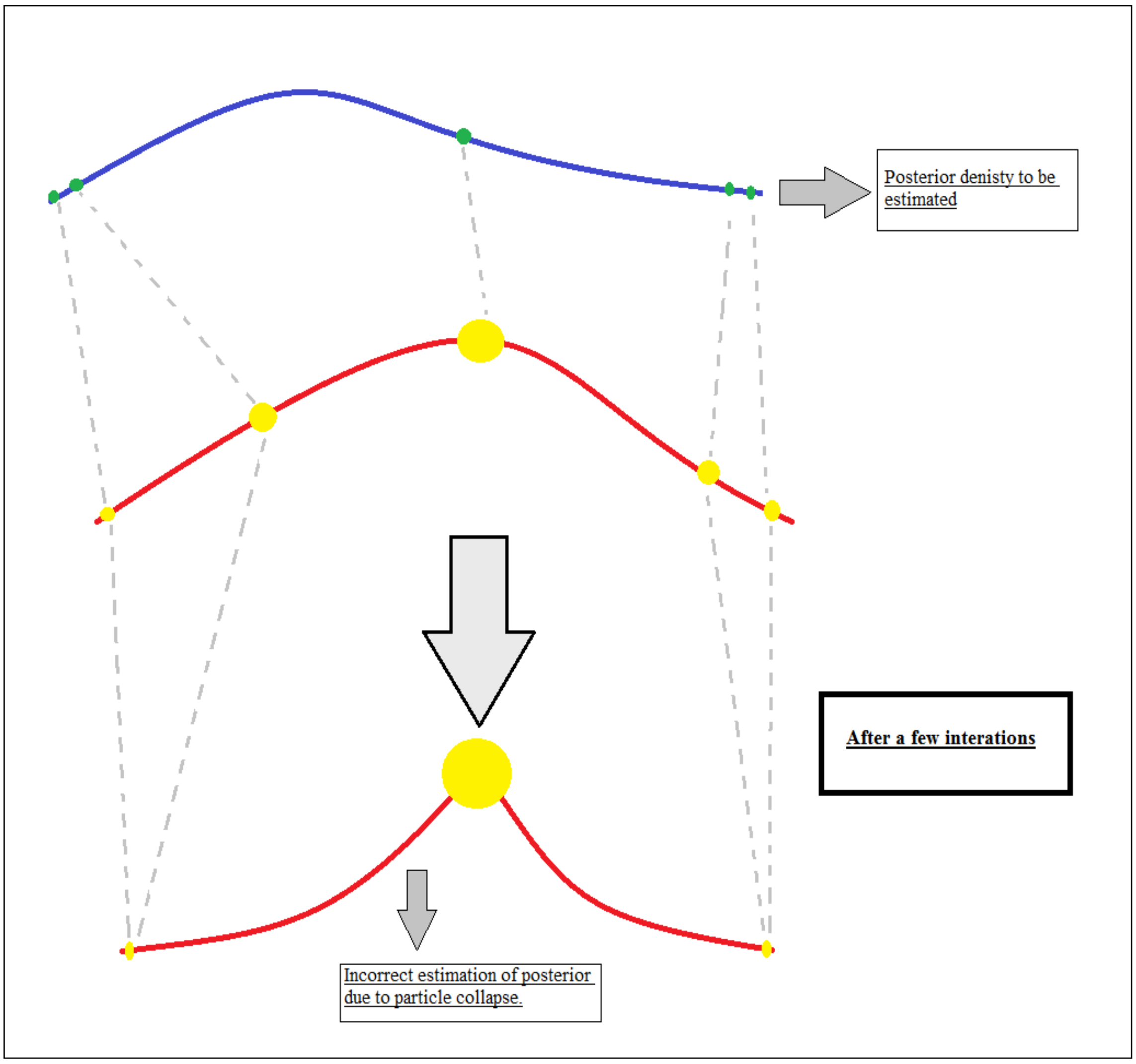

Posterior denisty to be estimated
After a few interations
Incorrect estimation of posterior due to particle collapse.

Figure 3-2: Incorrect Estimation in Particle Filters after a few Iterations

Increasing the number of particles can overcome this issue, however as the dimension of the problem is increased, it becomes even more difficult to avoid sample degeneracy and provide accurate estimates, as each particle needs to represent a sample from the posterior in all dimensions. The search space from which the particles need to be sampled has thus increased. Since this is not feasible, a particle filter needs to be efficient enough to search the space of all possible particles, and use the current weight of the particle to aid in the search process. This phenomenon is similar to premature convergence in evolutionary algorithms, a phenomenon noted in [RN99].

The resampling step introduced for the sample impoverishment problem will make copies of particles with high weight, but will not effectively search the space of particles. Conceptually, a particle population can be seen as a set of vectors, where each dimension of the vector represents a specific dimension of the state space. A particle can thus consist of components that efficiently sample from a particular dimension, and components that do not efficiently represent that dimension. The weight of a particle is thus a representation of all these components. The components that are responsible for contributing the most to a particle's weight can be seen as low-order hyperplanes.

The next diagram shows a particle population arranged in descending order of their weights. Each dimensions (shown as a square in the diagram) needs to be explored and searched across to be able to generate particles that represent the particle population correctly. Hence a single particle can be made up of components that correctly represent the sample in a particular dimension and components that do not provide a good estimate of the dimension they represent. A particle filter should be able to combine the 'good-components' onto a single string to be able to function in high-dimensions without suffering from ensemble collapse.

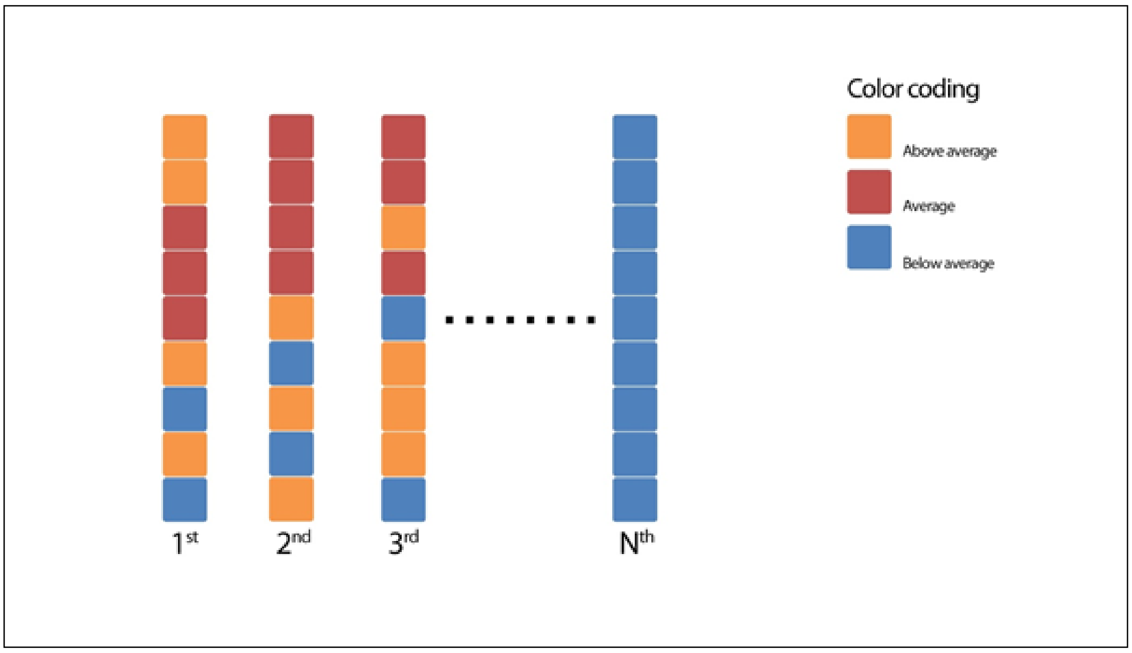


Figure 3-3: Particles Representing a High-Dimensional State space

A particle representing an n-dimensional state can be seen as a vector with n components. Consider a particle vector of dimension equal to five with above average fitness as shown in the figure 3.4:

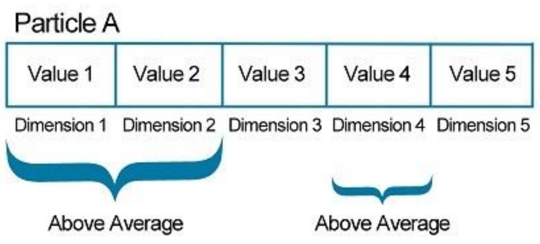


Figure 3-4: A 5-Dimensional Particle Vector

In this particle, the dimensions (components) 1, 2 and 4 contribute to the overall fitness (weight) of the particle; hence we can describe a vector template with only these values in those specific dimensions as shown in the next figure.

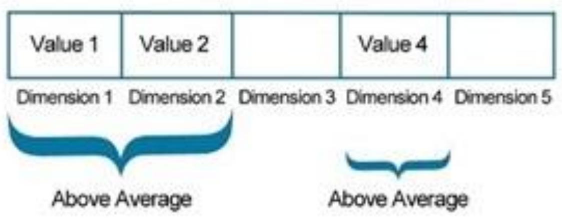


Figure 3-5: An Above-Average Fitness Hyperplane

The vector template shown in the above figure describes a desirable particle template. A template can thus be seen as a hyperplane. We thus have two main objectives to scale to higher dimensions, mathematically, if n(H(t)) is the number of hyperplanes of above average fitness at time t, then the particle filter should have a constructive ability, i.e., consider two short order hyperplanes H_k and H_l. The requirement is for the particle filter to be able to combine these two hyperplanes to create H_n, where n > k,l and is of above average fitness. Mathematically we can describe our second objective as:

$$E\left(n\left(H_n(t+1)\right)\right) \geq E\left(n\left(H_k(t)\right)\right) + E\left(n\left(H_l(t)\right)\right) \tag{3.1}$$

The requirements mentioned are similar to the theoretical approaches used in evolutionary computation to describe and predict the behaviour of genetic algorithms. Of particular importance for us would be Holland's Schema theorem, the building-block hypothesis and the role of recombination for constructing high-order hyperplanes.

Hence it can be concluded that the requirements to make particle filters scale to higher dimensions are similar to the theoretical foundations of GA that were used to explain their working.

It should be noted at this stage that a discrete based approach is used in this thesis to explain the concepts and workings of the RGAPF only, since it is easier to show in diagrams and that makes it easier to convey the concepts to the reader. The discrete recombination operators are not recommended for real-optimization problems as was mentioned in [BD01] and as the results of chapter 6 in this thesis will later show.

3.3 Adding Recombination and Mutation Operators in a Particle Filter

Thus far in this chapter the similarities between the particle filters and GAs have been established. The main difference is the absence of a recombination operator in a particle filter, which has been shown in GA literature to be solely responsible for the construction of higher-order hyperplanes i.e., the building-block hypothesis. Furthermore in section 3.2 it was argued that the main cause of sample impoverishment in higher dimensions is the inefficiency of a particle filter to explore the space of samples, which could be addressed if a particle filter was made to follow the building-block hypothesis.

It can thus be argued that the addition of a recombination operator, after resampling and artificial evolution, may be able to address the sample impoverishment in higher dimensions. It should be noted at this point that discrete values have been used in the diagrams to make it easier for the reader to imagine the working of a recombination operator in high-dimensions, in real domain, specialized real-recombination operators are used and provide better results compared to the n-point crossover.

The flow chart below shows the modification that needs to be made in a particle filter that may address the problems mentioned in this thesis.

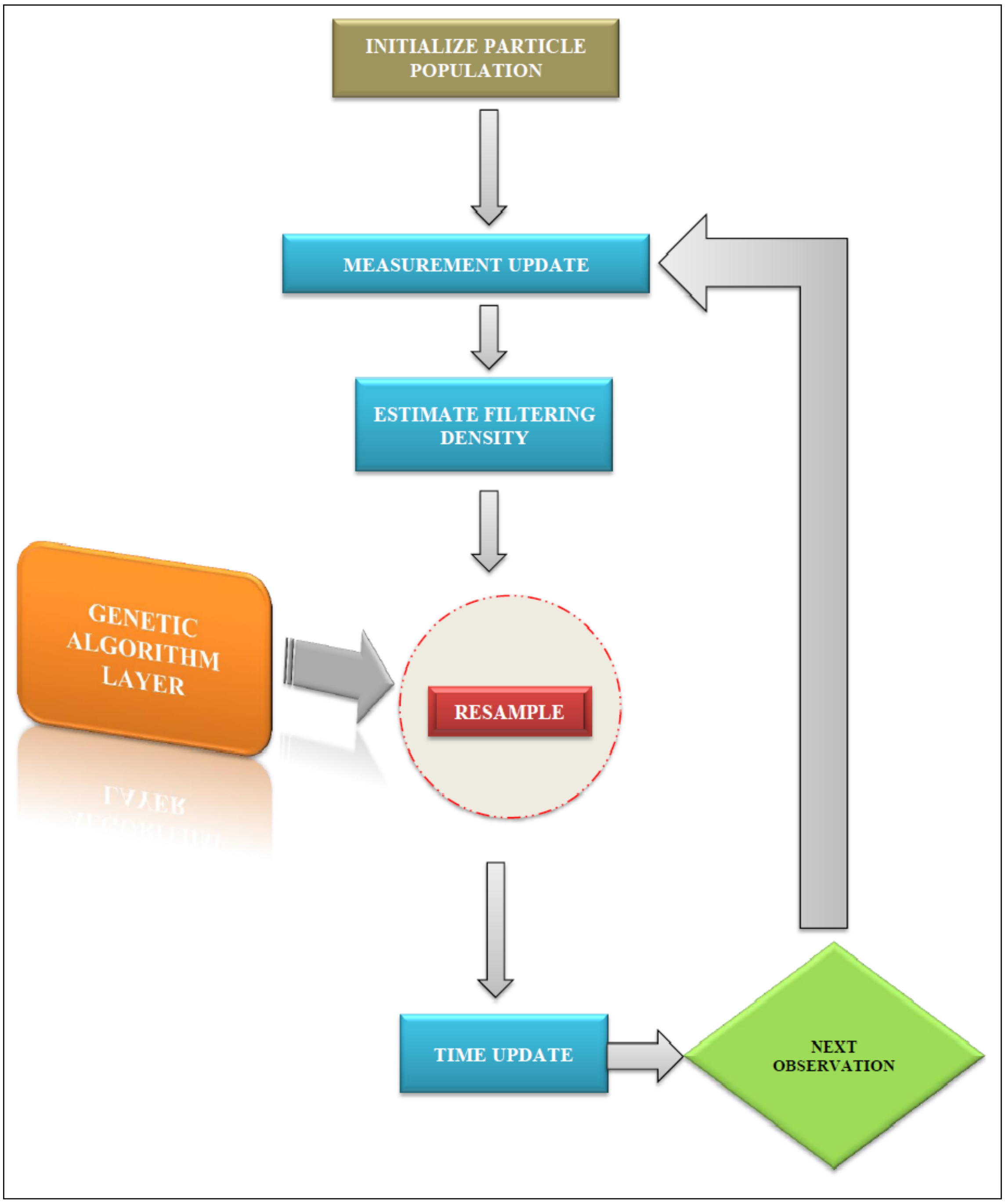


Figure 3-6: Addition of a GA layer in a Particle filter - A Flow Chart of the Required Algorithm

Consider figure 3.2 again. The reason for collapse of the particle filter was shown to be due to the inability of the particle filter to explore and search for particles. The addition of mutation and recombination may be able to address this issue. If our hypothesis is valid, then we might expect the particle filter with the GA layer to be able to function as shown in the next diagram.

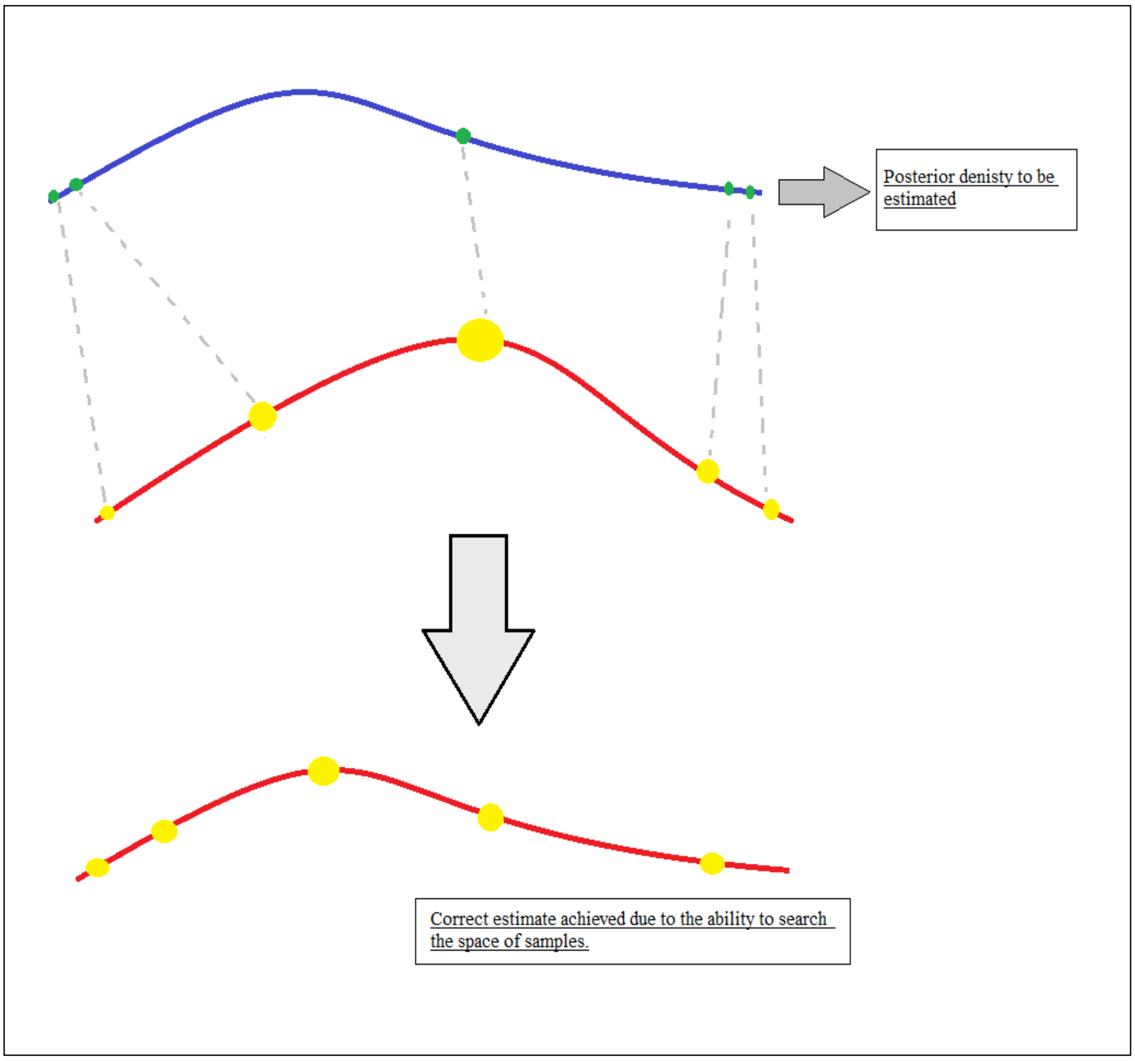

Posterior denisty to be estimated
Correct estimate achieved due to the ability to search the space of samples.

Figure 3-7: The Desired Outcome using RGAPF

The recombination operator may be able to combine the components that represent the posterior correctly onto single strings and hence be able to increase the efficiency of particle filters in higher-dimension.

The modified particle filtering algorithm, based on the concepts of genetic algorithms will be discussed in section 3.6 in detail.

3.4 The Real-Coded Genetic-Algorithm Particle Filter (RGAPF)

In the previous section it was concluded that the addition of a GA layer, comprising of a recombination operator and a mutation operator with a low mutation rate, in a particle filter may assist in the combination of building-blocks which could address the issues of sample impoverishments in high-dimensions. This section lays the foundation of the *real-coded genetic algorithm particle filter* (RGAPF). The RGAPF follows from the approach mentioned in the previous section and is used for experiments that are carried out in this thesis.

We start by describing the RGAPF, its pseudo-code and then carry out a comparison first with a generic particle filter, and then with the benchmark algorithms; the PLA and the PF-LW. We end with an over view of the different recombination operators that will be used in the experiments that follow in this thesis.

3.5 Recommended Modifications in a Particle Filter

In the previous section the particle filtering algorithm and the *genetic algorithm* (GA) were compared and it was shown that they are fundamentally similar. The main difference between the two algorithms was the presence of a recombination operator in a GA and a mutation operator with a very low mutation rate.

In GA literature, the recombination operator is credited for the ability to construct higher-order hyperplanes by combining lower-order hyperplanes. Holland's analysis suggests that selection increasingly focuses the search on subsets of the search space with estimated above-average fitness (defined by schemas with observed above-average fitness), whereas recombination puts high−fitness building blocks together on the same string in order to create strings of increasingly higher fitness. Mutation plays the role of an ‘insurance policy’, making sure genetic diversity is never irrevocably lost at any locus [Mit98].

We postulated that the ability of the recombination operator to put together higher-observed fitness building-blocks on the same string may in fact be able to address the collapse of particle filters in higher-dimensions. Based on these conclusions the following modifications need to be made in a particle filter:

1. Carry out selection (resampling) at each iteration.
2. Add a recombination operator after selection.
3. Add a Gaussian mutation operator after recombination.

Spears in [Spe98] observed that at a recombination rate of 0.5 maximum construction takes place. He also showed that the constructive ability of a GA reduces with an increase in the mutation operator. Keeping these observations in mind, we recommend that the RGAPF use a recombination rare of 0.5 and mutation rate of 0.02. The parameter values are kept constant even when the size of the population is varied. In [ES07], the authors mention the use of layer evolutionary algorithms to further optimize the parameter settings, however this is beyond the scope of this thesis. A flow chart describing the modified particle filter is shown in figure 3.9.

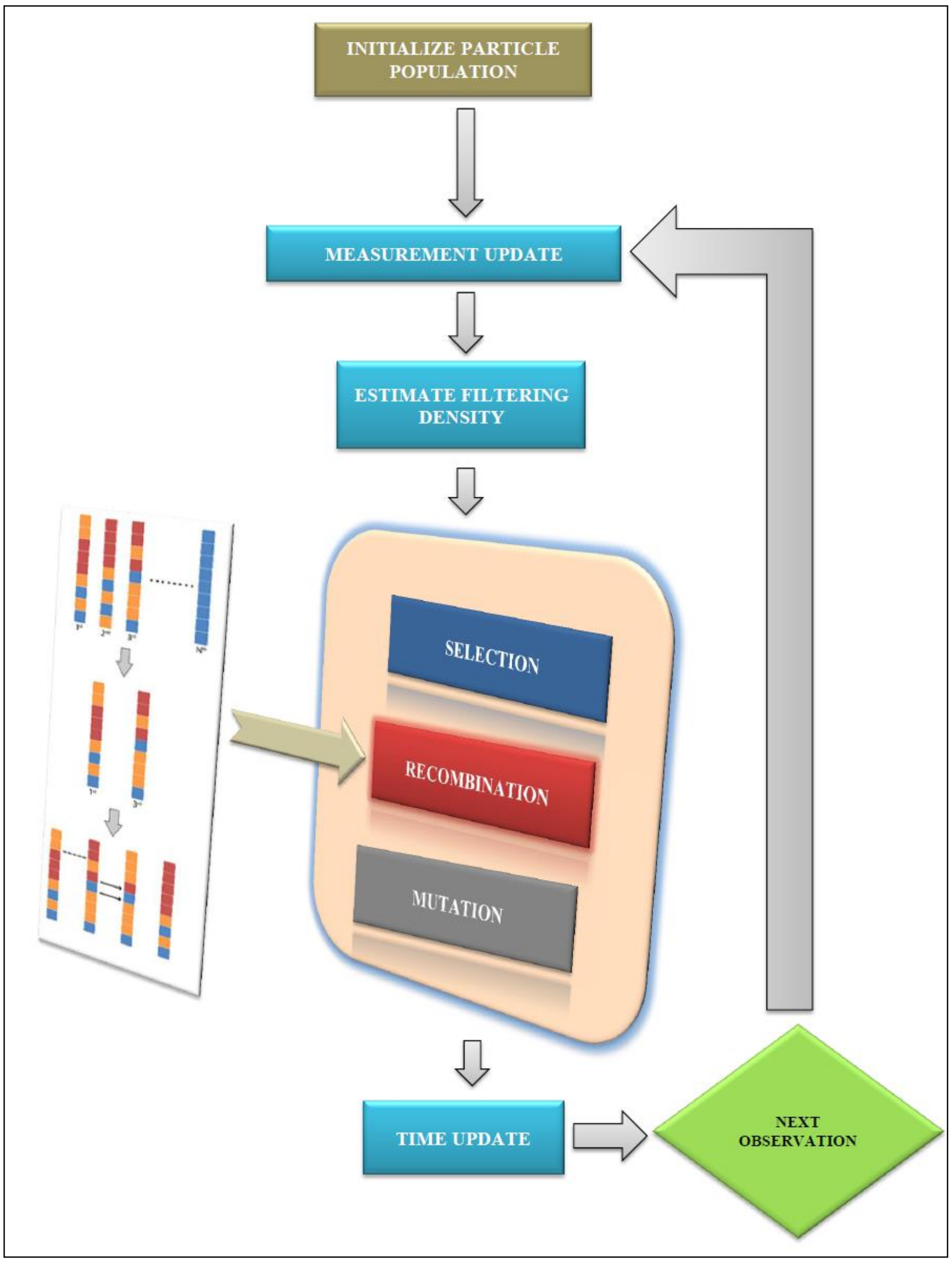


Figure 3-8: The Real-Coded Genetic Algorithm Particle Filter

Algorithm 3.2: The RGAPF – Pseudocode

```
Choose a proposal distribution $q(x_{k+1}^i|x_k^i, y_{k+1})$, resampling strategy and the number of particles N.
Initialization: Generate $x_1^i \sim p_{x_o}, i = 1, \dots, N$ and let initial weights to be 1/N.

For loop $k$ = 1, 2…End of Observations
START
    1. Measurement Update: For i = 1,2,…, N
           $w_{k|k}^i = \frac{1}{c_k} w_{k|k-1}^i p(y_k|x_k^i)$
       Where the normalization weight is given by:
           $c_k = \sum_{i=1}^N w_{k|k-1}^i \, p(y_k|x_k^i)$
    2. Estimation:
       The filtering density is approximated by
           $p(x_{1:k}|y_{1:k}) = \sum_{i=1}^N w_{k|k}^i \, \delta(x_{1:k+1} - x_{1:k+1}^i)$
       And the mean is approximated by:
           $\dot{x} \approx \sum_{i=1}^N w_{k|k}^i \, x_{1:k}^i$
    3. Time Update:
       Generate predictions according to the proposal distribution
           $x_{k+1}^i \sim q(x_{k+1}|x_k^i, y_{k+1})$
       And compensate for the importance weights
           $w_{k+1|k}^i = w_{k|k}^i \frac{p(x_{k+1}^i|x_k^i)}{q(x_{k+1}^i|x_k^i, y_{k+1})}$
    4. Using stochastic selection, sample parent particles
    5. Apply recombination with a recombination rate 0.5
    6. Apply mutation with a mutation rate of 0.02
    7. IF ($y_k$ is not last observation)
       $k = k + 1$
       Go to step 1.
         Else End For loop.
END
```

The text in red in algorithm 3.2 is the main difference between a generic particle filter and the RGAPF. The selection operator in a GA and the resampling step in a particle filter are similar, and hence not highlighted above.

3.6 Comparison between RGAPF and a Generic Particle Filter

A further comparison between the generic particle filter and the RGAPF is shown in Table 3.2. The text in red in the table shows the additions that have been made in a generic particle filter. It should be noted at this point that the resampling step and the selection operator are similar.

Table 3-2: The RGAPF and a Particle Filter – A Comparison

	Particle Filter	RGAPF	Notes
1	Initialize particle population	Initialize particle population	
2	Assign weights to particles using state update equation & importance sampling.	Assign weights to particles using state update equation & importance sampling.	For particle filters, this step contains both time and state update.
3	Carry out resampling.	Carry out selection.	Selection operator is similar to resampling, and is hence not highlighted.
4		Carry out recombination	Recombination rate is set to 0.5
5		Carry out Gaussian mutation.	The mutation rate is set to 0.02
6	If new observation is received, go to step 2.	If new observation is received, go to step 2.	

The next table, table 3.3 gives a comparison between the benchmark algorithms and the RGAPF. The three algorithms are relatively similar, however the main difference is the addition of recombination in an RGAPF, and the lower rate of the mutation operator. Both the benchmark algorithms add Gaussian noise at a rate of 1, while in case of an RGAPF, this rate is brought down to 0.02. The rationale behind this is the highly disruptive nature of a mutation operator, and the requirement for scaling to higher-dimensions is to maintain above-average fitness schemata and combine them using recombination.

Table 3-3: The RGAPF, The PLA and PF-LW – A Comparison

	RGAPF	PLA	PF-LW
1	Initialize particle population	Initialize particle population	Initialize particle population
2	Assign weights to particles using state update equation & importance sampling.	Assign weights to particles using state update equation & importance sampling.	Assign weights to particles using state update equation & importance sampling.
3	Carry out selection.	Carry out resampling.	Carry out resampling.
4	Carry out recombination	-	-
5	Carry out Gaussian mutation.	Carry out artificial evolution.	Carry out artificial evolution.
6	-	After a pre-defined number of iterations, carry out an MCMC step	-
7	If new observation is received, go to step 2.	If new observation is received, go to step 2.	If new observation is received, go to step 2.

3.7 Summary

This chapter starts by discussing the reasons of failure of particles in high-dimensions and using a set of diagrams shows that the requirement is to be able to construct high-order hyperplanes by using hyperplanes of low-order with above average fitness. It was later discussed that this concept is similar to the work done in evolutionary computation to describe the working of genetic algorithms. In genetic algorithms, the recombination operator has been shown to construct high order schemata using schemas of low-order. In the last section, it was proposed that adding a recombination and mutation layer after resampling may be able to address the dimensionality issues faced by particle filters.

The *real-coded genetic algorithm particle filter* (RGAPF) is proposed next. A GA layer is added in a generic particle filter and replaces the resampling step that is found in a generic particle filter. It is hypothesized that the addition of this layer will be able to address the issues faced by particle filters in most practical application, specifically the issues related to the ensemble collapse in higher dimensions.

Section 3.6 started by describing the working of a RGAPF by using pseudocode and a flow chart. The RGAPF is then compared with the particle filtering algorithms that will be used as benchmarks in this thesis. The comparisons between the PLA and PF-LW shows that the major difference is the addition of a recombination operator in the RGAPF. Another major difference is that in the RGAPF the mutation rate is set to 0.02, while in the benchmark algorithms the rate was set equal to 1. The rationale behind this choice of mutation rate is based on the destructive property of mutation. This chapter concluded with the description of real recombination operators that will be used in this thesis.

The next chapter describes the experimental methodology and discusses the setup of the simulations and how the results of the experiments will be displayed graphically and in tabular form for analysis.

Chapter 4

Experimental Methodology

This chapter describes the basic setup of the experiments carried out in this thesis. A total of five experiments are carried out in later chapters however they all follow a setup similar to the one described here.

This chapter can be divided into the following main sections:

- In section 4.1 the statistic used to compare the performance of the different particle filtering algorithms and the rationale behind using it is discussed.
- Section 4.2 describes the two main phases in our experiments. The particle scaling phase and the dimensional scaling phase.
- In section 4.3 the format of the output of the results and their graphical interpretation is discussed.
- This chapter concludes with a summary in section 4.4.

4.1 Performance Measure

Traditionally the *root mean square error* (RMSE) has been used to measure the performance of particle filtering algorithms [AMGC02] and we also use this statistic in this thesis for comparing the performance of the different filtering methods under changing population size and state-dimensions. The RMSE is given by the following equation:

$$RMSE = \sqrt{\frac{\sum_i (x_t - x_t^i)^2}{n}} \tag{4.1}$$

Where n is the number of observations, x_t is the correct value at time t and x_t^i is the value predicted by the i^{th} particle filter at time step *t*.

The use of RMSE as a performance measure follows directly from the proof of convergence of a particle filter given by Crisan et al., in [CD02]. Consider a probability density function P, and a particle population that represents this density given by P^N. They showed that the following holds true for sequential Monte Carlo methods:

$$E[(< P^N, \emptyset >, < P, \emptyset >)^2]^{1/2} \leq C_n \frac{|\emptyset|}{\sqrt{N}} \tag{4.2}$$

Here

$$< P^N, \emptyset > = \int \emptyset(x) \mu(dx) \tag{4.3}$$

And $||\emptyset|| = sup_x \emptyset(x)$, with $\emptyset$ a bounded measureable test function. Thus the equation shows that the RMSE converges to 0 as the number of particles N is increased. It is hence a natural choice to use the RMSE as a measure of the performance of the particle filter. In our experiments we scale the number of particles and the dimensions, and thus show that in a few cases the algorithm converges to the best predictive values a lot sooner than as shown by the above equation.

4.2 Experimental Phases

Each experiment follows the same sequence of performance evaluations that consist of two phases:

1. Particle Scaling Phase

2. Dimensional Scaling Phase

In the main execution of our experiments these phases are described by two functions that are called in a predefined sequence. A description of these phases is given in the next subsection. A flow chart describing the performance evaluations is shown in figure 4-1.

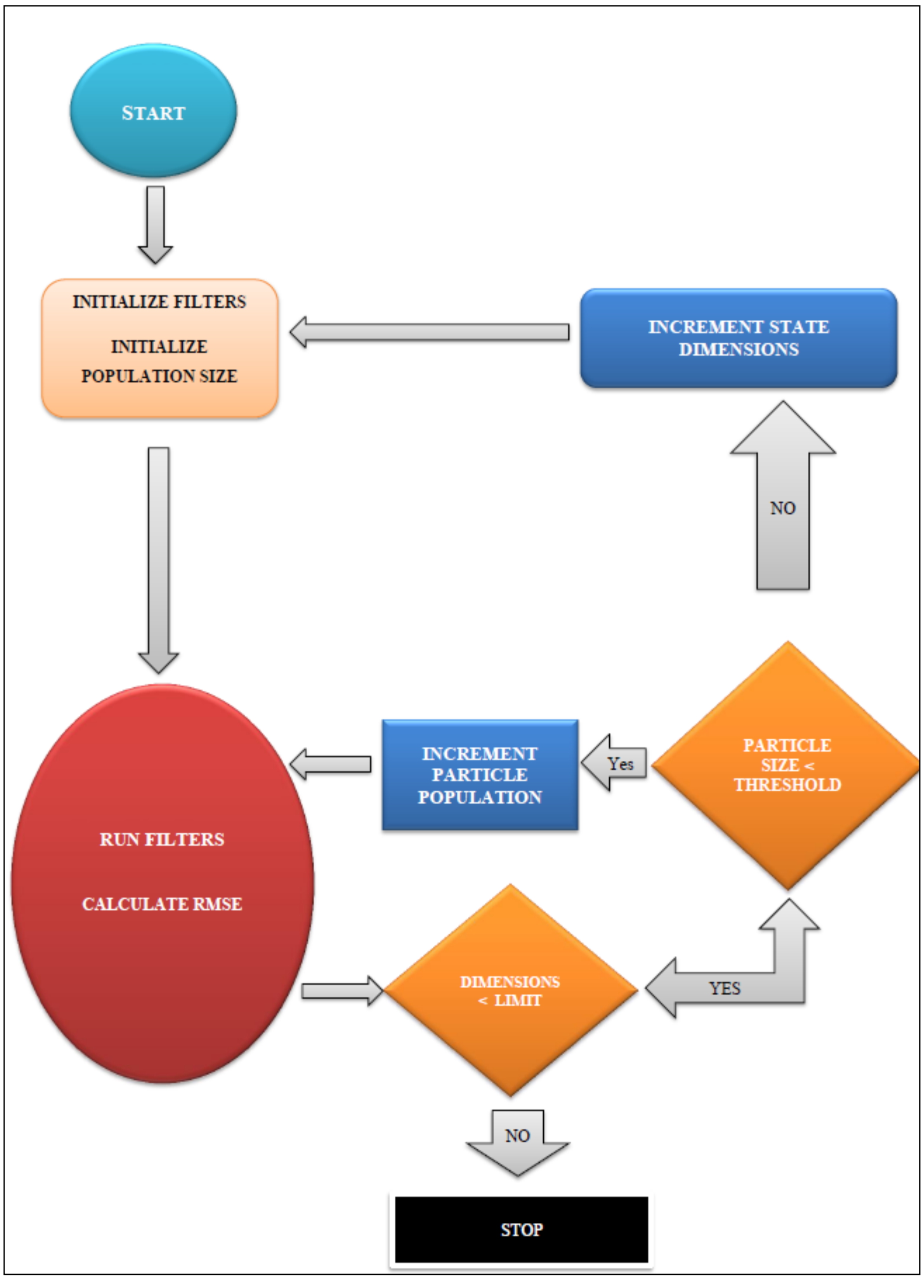

START
INITIALIZE FILTERS
INITIALIZE POPULATION SIZE
INCREMENT STATE DIMENSIONS
NO
RUN FILTERS
CALCULATE RMSE
INCREMENT PARTICLE POPULATION
Yes
PARTICLE SIZE < THRESHOLD
DIMENSIONS < LIMIT
YES
NO
STOP

Figure 4-1: Basic Setup of the Experiments

In the diagram above, the blue rectangles represent the main phases of our experiment. These two phases are described next.

4.2.1 Particle Scaling Phase

In this phase of the experiment, the number of particles of the particle filters are incremented for a comparison of the performance of each particle filters with increasing particles size. We start off with 10 particles, and run the particle filtering algorithms on the observation time series a total of 50 times to generate 50 RMSE values. An average of the RMSE calculated.

We then increment the number of particles and run the filters again a total of 50 times. This process is carried out for various increments in the particle size.

The second phase involves dimensional scaling, and is described in the next sub-section.

4.2.2 Dimensional Scaling Phase

In this phase of the experiment, the dimension of the state space is increased and the particle scaling phase described previously is carried out.

The stochastic volatility estimation problem is a 4 dimensional problem, with the dimensions being the state to be estimated and the three unknown parameters, i.e., k, ε, θ. To scale the dimensions further, an additional time series is provided to the filtering algorithms, hence with each additional time series, the dimensions get scaled by 4. Experiments carried out using simulated data are scaled to a maximum of 120 dimensions, while in the last experiment, with real time series, the dimensions are scaled to a maximum of 404. Hence for the different dimension and the particle sizes we generate the RMSE values for the filtering algorithms. The output of the two functions is thus a table of the following format:

[Particle Filter Name, Dimension Size, Particle Size, Average RMSE for 50 runs]

Table 4-1: Sample Performance Comparison Table

PARTICLE FILTER NAME	DIMENSIONS	No. of PARTICLES	RMSE
PARTICLE FILTER 1	4	10	0.5
PARTICLE FILTER 2	4	10	0.5
PARTICLE FILTER 1	4	20	0.6
PARTICLE FILTER 2	4	20	0.7
…	…	…	…

PARTICLE FILTER NAME	DIMENSIONS	No. of PARTICLES	RMSE
PARTICLE FILTER 1	120	10	2.1
PARTICLE FILTER 2	120	10	3.2
PARTICLE FILTER 1	120	20	1.5
PARTICLE FILTER 2	120	20	2.5

The observation process is simulated in the first four experiments, while the last experiment uses real time series from the London Stock Exchange. The process of simulating the time series is described in the next section.

4.2.3 Generating the Observation Series

Simulated Time series

Before we proceed with describing the pseudo-code for the simulated data, we describe the equations used to generate the time series of prices and stochastic volatilities. The stochastic volatility model consists of a coupled differential equation shown below:

$$\frac{dS}{S} = \mu dt + V^{1/2} dW^{(1)} \tag{4.4}$$

$$dV = \mu_V dt + \sigma_V V^{\alpha} W^{(2)} \tag{4.5}$$

Where α is a positive constant, and $W^{(1)}$ and $W^{(2)}$ can be correlated or uncorrelated Brownian motions. The above equations can be converted into their equivalent discrete form using the Laplace method. The discrete version of the SV model is given in equations 4.6 – 4.7.

$$ln\,\hat{X}\,(t + \Delta) = ln\,\hat{X}\,(t) - \tfrac{1}{2}\,\hat{V}(t)^{+}\Delta + \sqrt{\hat{V}(t)^{+}}Z_X\sqrt{\Delta} \tag{4.6}$$

$$\hat{V}(t + \Delta) = \hat{V}(t) + k\big(\theta - \hat{V}(t)^{+}\big)\Delta + \varepsilon\sqrt{\hat{V}(t)^{+}}Z_V\sqrt{\Delta} \tag{4.7}$$

Where $\hat{X}(t)$ is the observed price process and $\hat{V}(t)$ is the volatility process that has to be estimated.

For a dual estimation process the parameters k, ε, θ will also be estimated online along with the stochastic volatility. Starting with some initial parameter values for k, ε, θ, the volatility process is generated first and then the price process is generated.

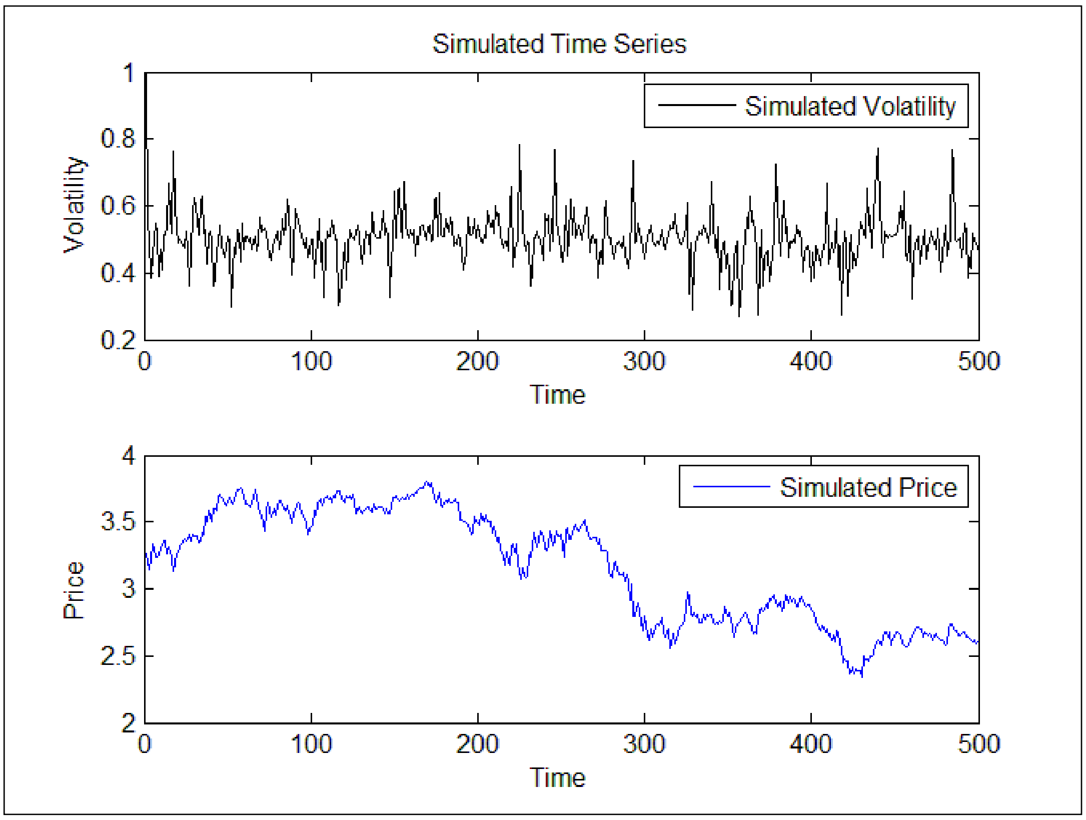


Figure 4-2: Simulated Time and Volatility Series

Real Data Set

For the experiment involving a real data set we use the asset prices that make up the FTSE-100 index, including the FTSE-100 index itself.

During the dimensional scaling phase, the dimensions are scaled by adding the next time series from the index. Hence the dimensions are scaled from 4 to 404.

4.3 Comparison of Results

In our experiments we are comparing different algorithms by changing the state-dimension and the number of particles. We have used a graphical approach for a better comparison of our results. We also provide abridged tables whenever we feel necessary, but due to the size of the tables they are not listed in the main text. When comparing the performance of two algorithms, the horizontal axis is used to represent the dimension of the state space, while the RMSE is represented by the vertical axis.

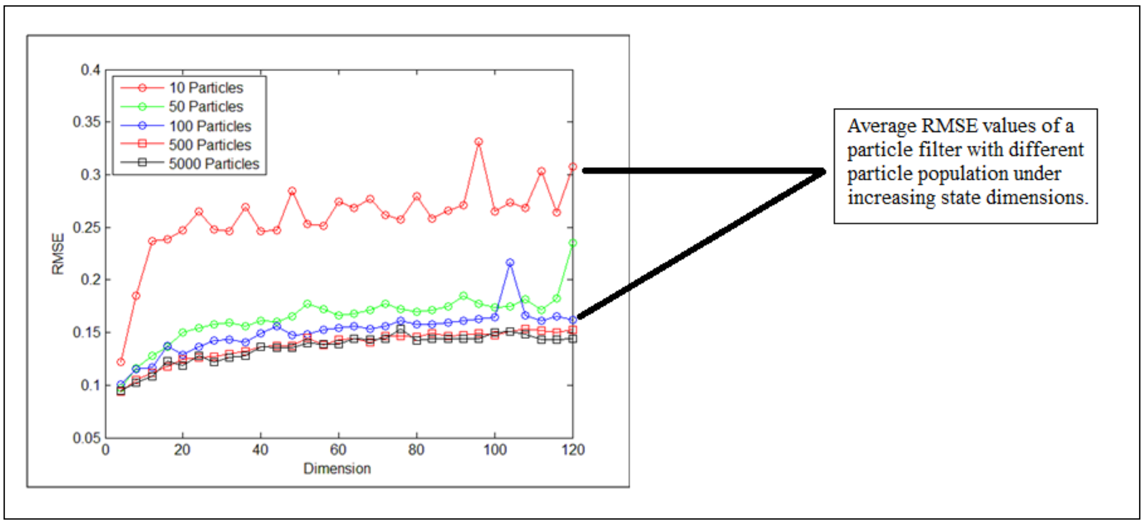


Figure 4-3: Comparison of Performance - Particle & Dimensional Scaling

The RMSE values are compared using ANOVA. When necessary, the p-values are also listed next to the *root mean square error* (RMSE) values in the comparison tables. A p-value less than 0.05 is used as a benchmark that signifies that the algorithms under comparison provides RMSE values that appear to be taken from different samples. Hence, in the text when two algorithms are mentioned to be ‘different’, then they have a p-value less than 0.05, while the term ‘significantly-different’ is used when the p-values are less than 0.001.

4.4 Summary

This chapter described the experimental methodology that will be used to carry out experiments in the next chapters. The proceeding experiments carry out the scaling of the number of particles and the dimensions of the state. Two functions are used for this purpose and they are described in this chapter. The experiments carried out in this thesis use a simulated observation process while the last experiment uses real data series. The simulated time series is generated using the Heston model and the process of simulation is described in section 4.2. The results obtained after carrying out the experiments are compared graphically; the format of these graphical representations is described in section 4.3.

In the next chapter, the first of our series of experiments is carried out. The experiment involves a comparison in the performance between the benchmark algorithms and the RGAPF under scaling particles and dimension size.

Chapter 5

RGAPF Performance Under Particle and Dimensional Scaling

In this chapter a performance comparison is carried out between the RGAPF and the two benchmark algorithms, *the particle filter of Liu & West* (PF-LW) and the *Particle Learning Algorithm* (PLA). The RMSE values of these algorithms are recorded while increasing the size of the particle population and the state-dimensions. The main observations of the experiment were as follows:

Low-Dimensional State Space.
The three filtering algorithms provide a similar approximation of the density function. The RMSE values obtained are similar, however the RGAPF converges in performance with less number of particles compared to the other filters.

High-Dimensional State-Space.
As the dimensions of the state are increased, the benchmark algorithms suffer from ensemble collapse. Increasing the number of particles has no significant effect on bringing the approximation error down. The RGAPF however is stable and continues to perform.

This chapter can be divided into the following three main sections:

- The first section, section 5.1, is divided into two parts. The first part compares the effect on the RMSE of the three algorithms when the number of particles is increased. The next part of this section tests the scalability of the filtering algorithms to higher dimensions.
- In section 5.2 we discuss the results of the experiments using concepts borrowed from GA theory.
- This chapter concludes with a summary in section 5.3.

5.1 Experiment 1- Performance Comparison with Benchmark Algorithms

In [CD05], it was mathematically shown that the prediction error of a particle filter decreases with an increase in particle population. Theoretically the prediction error approaches zero as the population size reaches infinity. In [SBBA08, BBL08] it was shown that the number of particles need to increase exponentially as the state dimensions are increased. This requirement has hindered the wide spread use of particle filters in most practical applications.

The next two subsections describe the experimental setup and the results of the experiment.

5.1.1 Particle Scaling

The first comparison is carried out keeping the state-dimensions constant at four and increasing the particle population. We start with 10 particles, and run the three algorithms on a simulated price time-series a total of 50 times and the average of the RMSE are calculated. The number of particles is then increased to 20, and the same procedure is carried out. The particles are then increased 10 at a time till we reach 100. After reaching 100 particles, the number is increased a 100 at a time till the population size is 600. We then increase the number of particles to a 1000, after which in each iteration the size is increased a 1000 at a time, till we reach 5000.

It should be noted at this point that for each population size, the simulation is run 50 times and the average of the RMSE is calculated.

The graph in figure 5.1 summarizes the results of this comparison. We can see that as the number of particles is increased, the performance of the three algorithms improves and becomes comparable after we reach 2000 particles. Increasing the number of particles further has no significant effect on the performance. However we can also observe that the PLA and

RGAPF converge to their optimal performance level for this dimension with lesser number of particles compared to the PF-LW.

The RGAPF uses arithmetic recombination given by equation 4.8. Where α is equal to 0.7 and the recombination rate is set equal to 0.5. A zero-mean Gaussian mutation operator is used with a variance of 0.15. The mutation rate is set equal to 0.02. These parameters are based on a similar experiment carried out in [SH12].

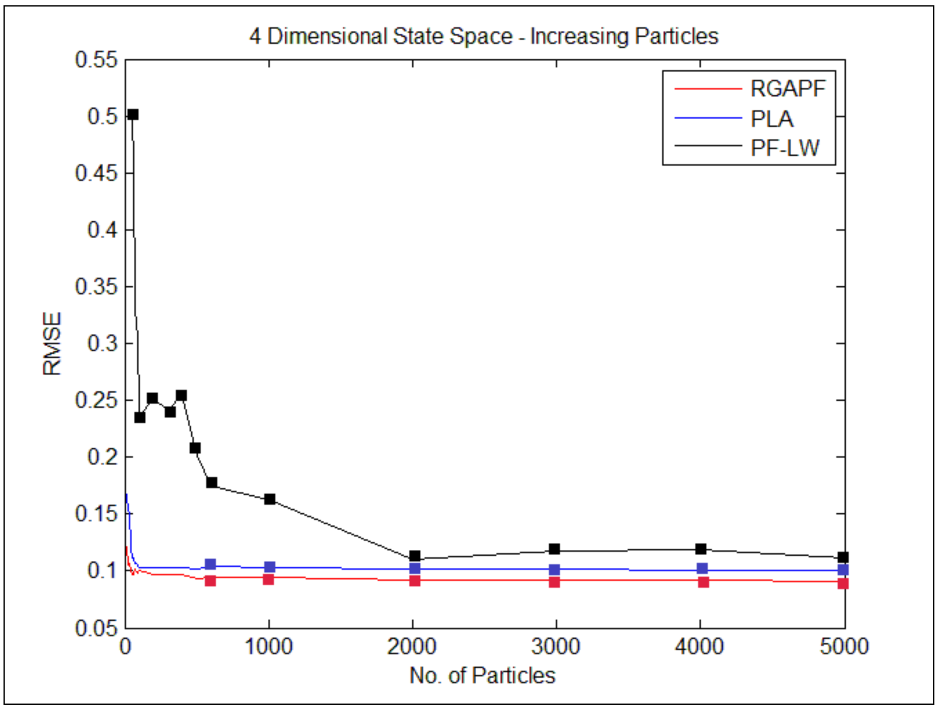


Figure 5-1: Effect of Increasing Particle Population on Performance

The PLA and RGAPF perform relatively well when compared to the PF-LW. The PF-LW requires a larger population size to converge. The RGAPF and PLA however give better estimates of the posterior even when the population size is quite low.

The graph in figure 5.2 compares the performance of PLA with RGAFP when the population size was 100 or less. Since the high RMSE values of the PF-LW estimates made it visually difficult to observe the performance of the other two algorithms, a logarithmic scale is used. We can see that the RGAPF gives a good estimate with lesser number of particles and converges quickly; however its performance is comparable to the PLA in a four-dimensional estimation problem. The PF-LW however requires a higher number of particles to provide this level of performance. As is evident in table 5-1, the standard deviation of PF-LW is orders of magnitude higher than the mean when a small population size is used. This observation follows directly from the conclusions mention in [LW01].

It can be concluded at this point that when the state-dimensions are low, the three algorithms can be made to provide an equivalent performance by adjusting the population size of the particles. In this particular experiment, for a population size of 600 or above, the three algorithms provided similar results (p-values > 0.05). It remains to be seen how they performance of these algorithms would be effected by the change in state-dimensions. The performance under dimensional scaling is carried out in the next subsection.

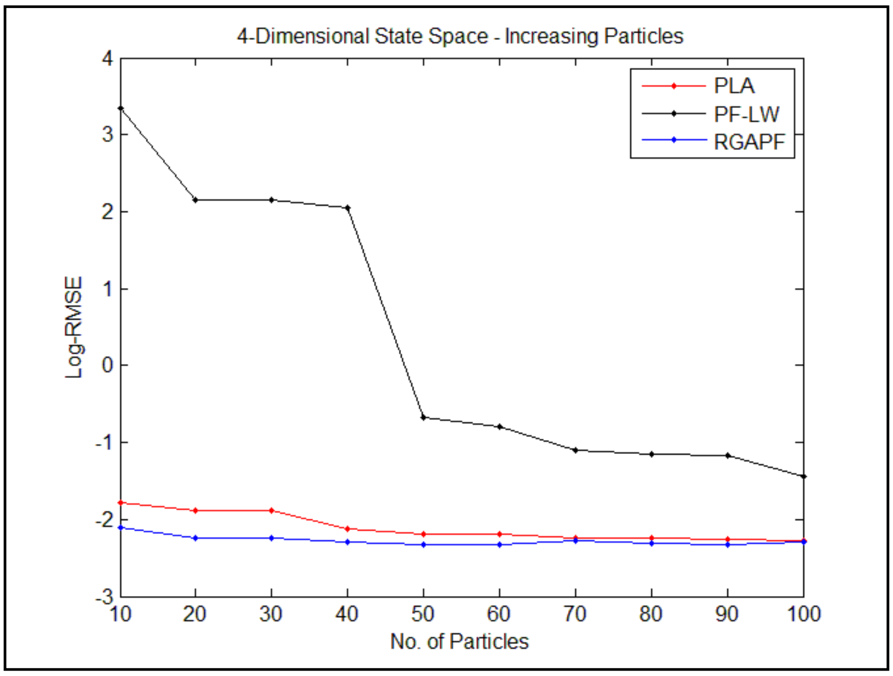


Figure 5-2: PLA vs. RGAPF - 4 dimensional State-Space

Table 5-1: Particle scaling in PLA, PF-LW and RGAPF

No. of Particles	PLA - RMSE	PLA – Standard Deviation	PF-LW RMSE	PF-LW Standard Deviation	RGAPF - RMSE	RGAPF – Standard Deviation	P-values
10	0.167702	0.023702	28.558101	89.41867	0.121634	0.023518	0.0006
20	0.153177	0.019707	8.650971	0.185981	0.105734	0.013201	0.0000
30	0.151688	0.123184	8.603219	25.92898	0.106121	0.009381	0.0000
40	0.120641	0.015283	7.715082	0.099539	0.100909	0.012544	0.0000
50	0.111863	0.00738	0.510102	1.293670	0.096883	0.008174	0.0000
60	0.111834	0.058864	0.452546	0.105238	0.097102	0.006303	0.0000
70	0.106114	0.013898	0.332279	88.06518	0.101844	0.015761	0.0001

No. of Particles	PLA - RMSE	PLA – Standard Deviation	PF-LW RMSE	PF-LW Standard Deviation	RGAPF - RMSE	RGAPF – Standard Deviation	P-values
80	0.106071	0.004793	0.315209	0.065861	0.098929	0.008136	0.0031
90	0.105058	0.023535	0.310125	209.2697	0.098365	0.006732	0.0024
100	0.102741	0.006268	0.235127	0.093910	0.100938	0.007631	0.0019
200	0.103464	0.007131	0.251041	0.065309	0.097327	0.009290	0.0008
300	0.102479	0.006077	0.241179	0.374141	0.096865	0.005496	0.0012
400	0.102483	0.007185	0.255446	0.748081	0.096955	0.005301	0.0101
500	0.101728	0.003376	0.209814	0.102627	0.093442	0.005151	0.0115
600	0.104494	0.107307	0.178553	0.171897	0.094317	0.004194	0.1071
1000	0.102842	0.004431	0.161924	0.344637	0.094762	0.002985	0.2643
2000	0.101924	0.004431	0.111003	0.344637	0.091907	0.002985	0.7697
3000	0.101991	0.004431	0.115171	0.344637	0.092082	0.002985	0.2021
4000	0.100749	0.004431	0.113794	0.344637	0.091791	0.002985	0.3334
5000	0.100505	0.004431	0.114102	0.344637	0.090991	0.002985	0.9803

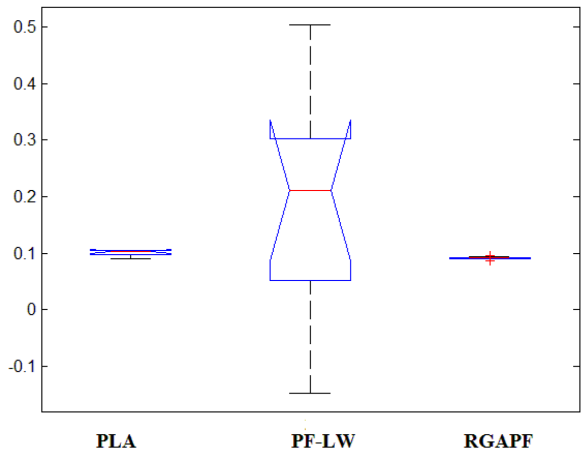


Figure 5-3: Box-Plot – PLA, PF-LW & RGAPF – 4-Dimensions

5.1.2 Dimensional Scaling

We next test the performance of the three algorithms as we increase the dimensions of the state space. The experiment is similar to the previous experiment, and the same number of particles is used at each step, however after every 50 evaluations, another price time series is added to the problem, which increases the dimension of the state by 4. This procedure is carried out till the dimension of the state reaches 120 (i.e., 30 price time-series).

Eight Dimensional State-Space

The second iteration of the dimensional scaling comparison increases the dimension of the state to eight.

The next two graphs compare the performance of the PLA and the RGAPF at this stage of the comparison. The performance of the PF-LW had deteriorated significantly and has thus been omitted in the comparison (p-value < 0.001). Both the PLA and the RGAPF provide a similar level of performance. The RGAPF converged earlier requiring lesser number of particles, however as the number of particles is increased, the performance of the two appears similar with no major difference between the two.

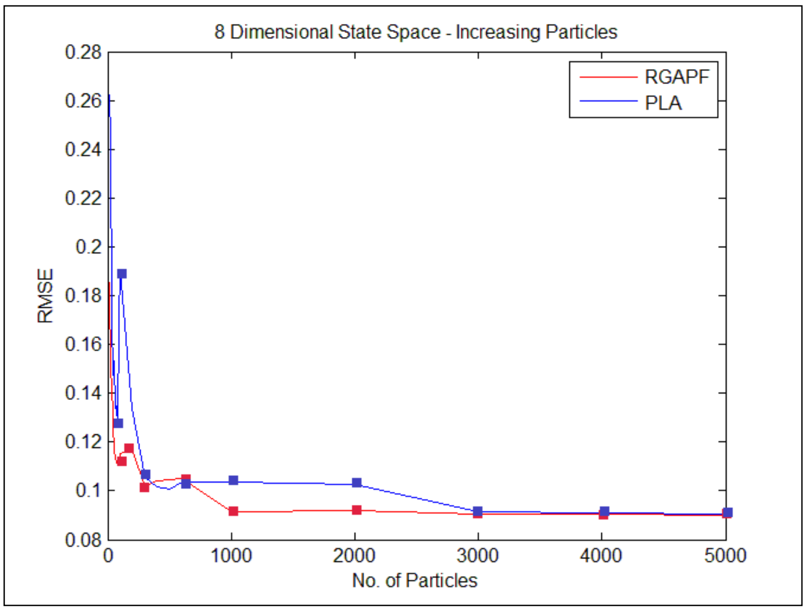


Figure 5-4: PLA vs. RGAPF – Particle Size 10 - 5000

Twelve Dimensional State-Space

On the third iteration of the dimensional scaling phase the state dimensions reaches twelve. At this stage the PLA performance starts to show signs of deterioration. The PLA is unable to effectively estimate the posterior when the population size is low, however increasing the number of particles does improve its performance but it is unable to match the performance it achieved when the state-dimension were low.

It can be seen in the graphs that the PLA performance starts to deteriorate after the state-dimensions are increased beyond eight. It can be argued, looking at this graph, that increasing the dimensions further would affect the PLA performance severely; and this was our observation once the state-dimension reached sixteen (the fourth iteration of the dimensional scaling phase).

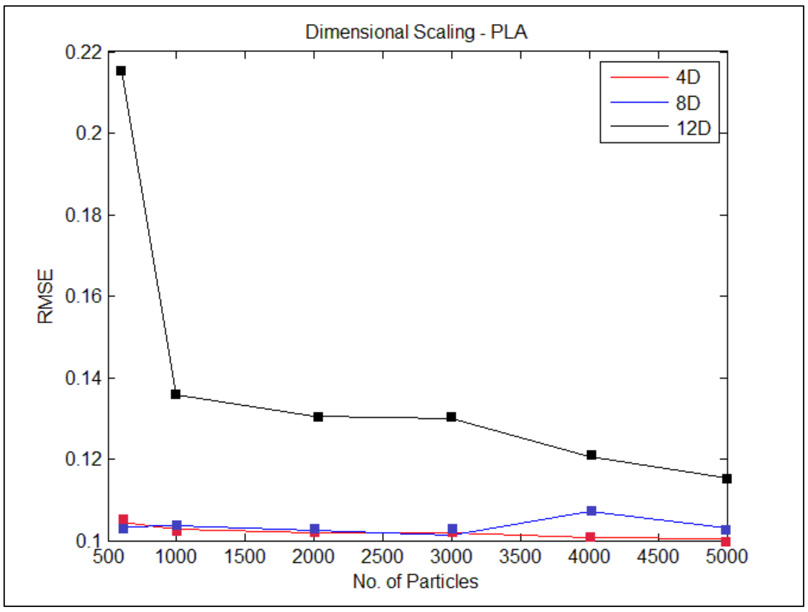


Figure 5-5: The PLA Performance under Dimensional Scaling

We plot similar graphs for the RGAPF; however the RGAPF is resistant to the increase in dimensions and still provides reasonable estimates as the number of dimensions are increased. The graph in figure 5-6 show the performance of the RGAPF with increasing dimension and particle population. Compared to the bench mark algorithms, the RGAPF is able to scale to higher-dimensions. The graph in figure 5-6 does not include error bars as the standard-deviations are listed in table 5-3, and these are of the order of 0.01 or less.

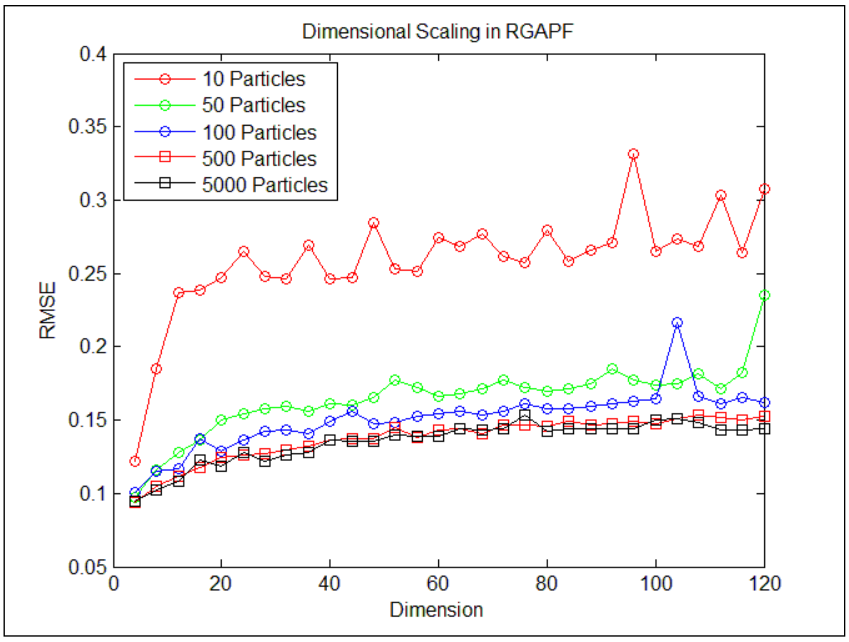


Figure 5-6: Dimensional Scaling in RGAPF – Dimension on the Horizontal Axis

These results demonstrate that the RGAPF converges to a good estimate with lesser number of particles and increasing the number of dimensions does not have any significant effect on the performance. The results also show that the RGAPF converges with less number of particles and increasing the number of particles considerably is not required. The hypothesis and approach of this thesis seems valid after going through these results. Table 5.2 lists the complete results of our experiments for an RGAPF with an arithmetic recombination operator.

Table 5-2: Effect of Increasing dimensions and particles on the performance of RGAPF

No. of Particles	Dimension	RGAPF – Arithmetic (RMSE)	Standard Deviation
200	4	0.094798	0.00693
200	8	0.105555	0.00644
200	12	0.112767	0.005629
200	16	0.116657	0.004694
200	20	0.124485	0.008702
200	24	0.135673	0.010484
200	28	0.13221	0.011942
200	32	0.135221	0.007907
200	36	0.145563	0.015931

No. of Particles	Dimension	RGAPF – Arithmetic (RMSE)	Standard Deviation
200	40	0.134765	0.00564
200	44	0.142544	0.005012
200	48	0.142253	0.00699
200	52	0.149983	0.014409
200	56	0.145782	0.010043
200	60	0.145487	0.006051
200	64	0.143132	0.008607
200	68	0.149438	0.008884
200	72	0.144875	0.003727
200	76	0.151105	0.010951
200	80	0.146807	0.00552
200	84	0.147947	0.006285
200	88	0.152669	0.006102
200	92	0.153768	0.010361
200	96	0.157221	0.011392
200	100	0.147865	0.002091
200	104	0.152966	0.007475
200	108	0.157882	0.010289
200	112	0.16035	0.00775
200	116	0.155681	0.004655
200	120	0.153326	0.007498

5.2 Discussion

Two main observations of the experiment carried out in the previous section are:

- The RGAPF is able to scale to higher-dimensions.
- Once the RGAPF performance has converged, increasing the number of particles further has no significant effect on the RMSE of the estimates.

The observations can be explained using the concepts of the building-block hypothesis and implicit parallelism.

5.2.1 Scalability to Higher-Dimensions via Schema Construction

The objective of the particle filter is to assign weights to the particles that provide a good estimate of the posterior; the particle weights are updated after every arriving observation however the diversity of the particles is limited by the initialized particles. Throughout the filtering process no new particle is created. In high-dimensional cases the particle filter is thus required to be initialized with a very large number of particles so as to provide it with a reasonably sized search space. Both the PF-LW and PLA use an operator similar to an annealed mutation to add diversity to the particles, however their search operator is unable to search efficiently in high-dimension. The scalability of the RGAPF on the other hand can be explained using the constructive property of a recombination operator.

5.2.2 Convergence to Posterior using Less Number of Particles – Implicit Parallelism

Another important observation was that the improvement in the RGAPF performance when the number of particles is increased is gradual and the performance of the RGAPF with a 100 particles is similar to when 5000 particles were used. This observation can be explained using the concept of implicit-parallelism. Holland in [Hol75] showed that a string of length $\boldsymbol{l}$ is an example of 2^l schemata and in a population of μ parents the GA will usefully process $O(\mu^3)$ schemata. According to this phenomenon, while the algorithm may be explicitly sampling and evaluating a smaller number of parents, implicitly it is sampling from a much larger population set. This is known as implicit-parallelism and is quoted as one of the main reasons of the success of genetic algorithms [Mit98].

Hence in the graph describing the RGAPF performance, the improvement in the approximation is slightly increased when the numbers of particles are increased from experiment to experiment.

5.3 Summary

In this chapter the RGAPF was compared with the PLA and the PFLW under varying state-dimensions and particle population. The RGAPF is able to scale to higher dimensions unlike the other two algorithms.

Two observations were made when the results of the RGAPF were analysed. Not only is the RGAPF able to perform in a high dimensional state-space, it also requires less number of particles for a good estimate of the posterior. The improvement in performance when the numbers of particles were increased was only slight. This led us to conclude that this may be due to the property of a GA where the algorithm may be explicitly sampling and evaluating a smaller number of parent particles, while implicitly it is sampling from a much larger population set. The scalability to higher dimensions was explained via the building-block hypothesis and the constructive property of a recombination operator. These results follow from the hypothesis and approach of this thesis where we observed that due to the similarities of the particle filter and the GA the addition of GA operators, specifically recombination, will be able to address the issue of sample impoverishment in higher dimensions.

In this chapter we have used the building-block hypothesis to explain the success of the RGAPF in higher dimensions. Since the building-block hypothesis credits the recombination operator for its ability to combine low-order schemata to create schema of high-order and high-fitness, to further strengthen our argument about the importance of recombination in high dimensional scenarios the next chapter carries out an experiment where a mutation only particle filter, with an adjustable mutation rate, is compared with the RGAPF.

Chapter 6

The Role of Recombination in High Dimensional Particle Filtering

Results of the previous chapter demonstrate that the addition of a genetic algorithm layer enhances the performance of a particle filter and enables it to scale to higher dimensions. The approach of this thesis is based on the building-block hypothesis which credits the recombination operator as being responsible for the construction of highly-fit schema in genetic algorithms. To further strengthen our argument that building-block hypothesis like effects are due to the recombination operator, we carry out an experiment in this section that runs two particle filters in parallel. The first particle filter has recombination and mutation, while the second particle filter has only Gaussian mutation. To test whether recombination is contributing to building-block like effects or simply acting as an effective variable-rate mutation operator, we “tether” the mutation rate of the mutation-only particle filter to the effective population-to-population variance in the recombination-plus-mutation particle filter. The latter significantly and consistently performs better, indicating that recombination is having a subtle and significant effect that may be theoretically explained by genetic algorithm theory.

The results of this experiment further strengthen our belief in the hypothesis that the addition of a recombination operator introduces building-block like effects in a particle filter that helps address the phenomenon of sample impoverishment in higher dimensions. We then move on to test the performance of the RGAPF with different recombination operators. The RGAPF with the mean-centric recombination operator, UNDX, provides the best estimates of the posterior.

This chapter is divided into the following four main sections:

- Section 6.1 recounts and summarizes the conclusion of the previous chapter.
- The design, results and discussion of the experiment that tests the constructive ability of recombination is given in section 6.2.
- A comparison between the performances of the RGAPF using different recombination operators is carried out in section 6.3.
- The chapter concludes with a summary in section 6.4.

6.1 The Role of Recombination in Constructing High-Order Hyperplanes

Particle filters have been known to collapse in high-dimensional scenarios however the results of the previous experiment show that the addition of recombination addresses this issue. The approach mentioned in chapter 3 and the discussion at the end of the last chapter used the schema theorem and the building-block hypothesis to explain how dimensional scaling will take place in a particle filter once a recombination operator is added.

The genetic algorithm has two main search operators, the recombination operator and the mutation operator. The recombination operator is credited for the building-block like effects in a GA, whereas the mutation operator adds diversity and ensures that the GA does not suffer from hitch-hiking [MHF94].

6.1.1 Recombination vs. Mutation

Apart from Genetic algorithms, evolution strategies and genetic programming are the two other principal forms of evolutionary algorithms [Tal01]. Evolution strategies are optimising methods very similar to genetic algorithms. Evolution strategies differ from genetic algorithms in two aspects. Evolution strategies use real values to represent gene organisms instead of encoded strings. However, more importantly, evolution strategies focus mainly on the mutation operator [Tal01].

Both mutation and recombination add diversity to the candidate population and perform search in the search-space, however their mechanism of action is completely different. The recombination operator has been credited for the adaptive nature of GAs [Hol75] while the mutation operator was added as an insurance policy to aid in diversity [Mit98].

Fogel et al., in [FA00] stated that recombination is a generalization of several mutations performed at once. However Spears in [Spe98] showed that the construction of high-order hyperplanes took place because of the recombination operator and that maximum construction occurred when the recombination probability was equal to 0.5. He also showed that the disruption increases with the increasing mutation rate. The hypothesis of this thesis is based on the constructive ability of the recombination operator.

In the next section we carry out an experiment that tries to test whether recombination does in fact combine building-blocks and whether a similar behaviour can be achieved by using mutation only.

6.2 Experiment 2 - Construction of High-Order Hyperplanes in a particle filter

For an effective comparison between recombination and mutation we run two algorithms in parallel. The experimental setup is similar to the previous chapter, however there is a slight modification in the order in which the two algorithms are run, i.e., after the arrival of each measurement from the observation series, the RGAPF updates the posterior and then the population variance change is calculated. The second particle filter, uses this statistic to adjust its mutation rate. Both the algorithms use Gaussian mutation with zero mean and variance equal to 0.15, however the mutation-only particle filter carries out mutation on its population until its population-to-population variance reaches the value provided by the RGAPF. The other parameter values in this experiment are similar to the parameter values used in the experiment described in section 5.1.2.

Flow charts describing the working of the algorithms are shown next.

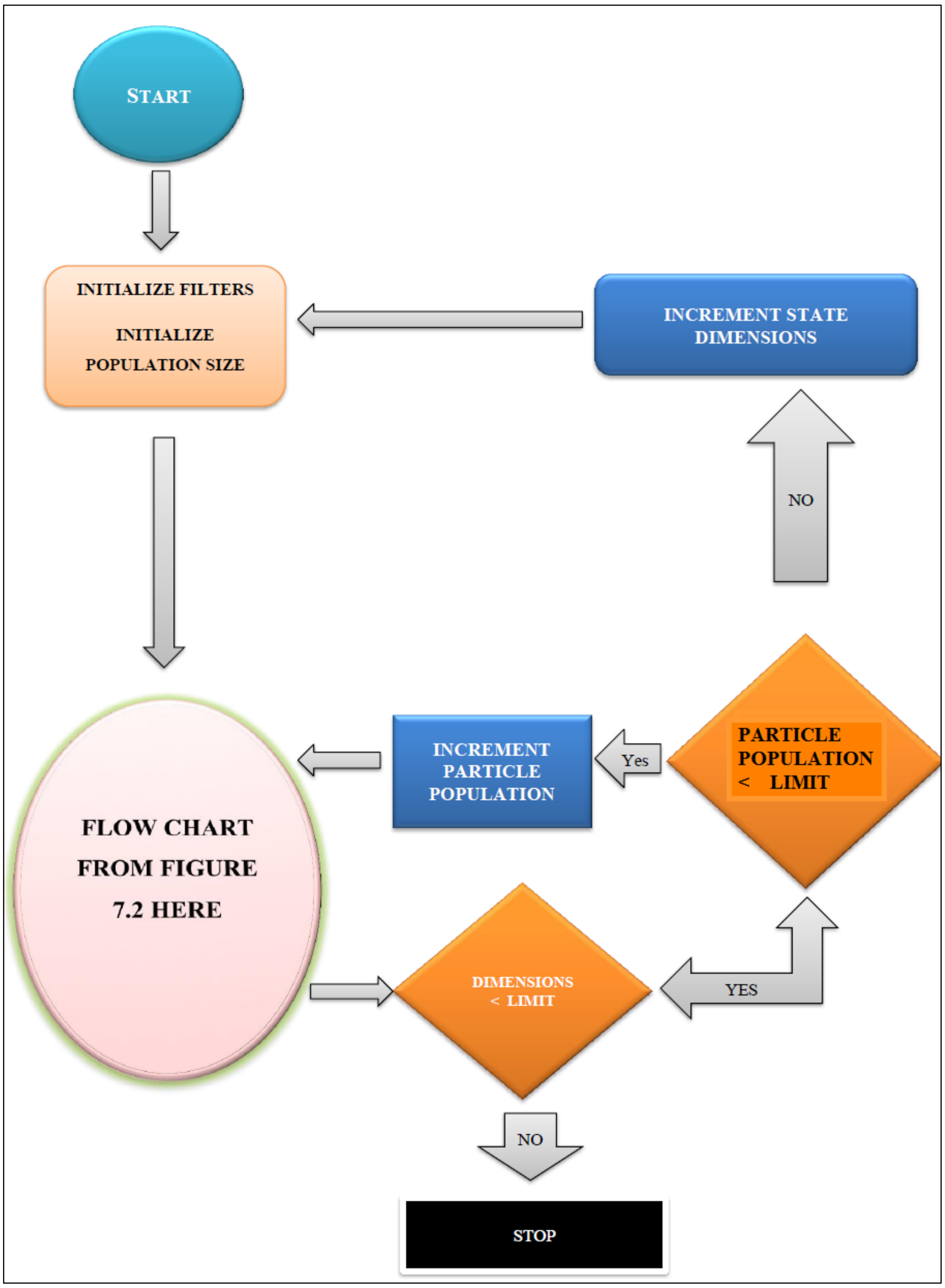

START
INITIALIZE FILTERS
INITIALIZE POPULATION SIZE
INCREMENT STATE DIMENSIONS
NO
FLOW CHART FROM FIGURE 7.2 HERE
INCREMENT PARTICLE POPULATION
Yes
PARTICLE POPULATION < LIMIT
DIMENSIONS < LIMIT
YES
NO
STOP

Figure 6-1: Design of Experiment - Part 1 (Scaling State Dimensions and Particle Population)

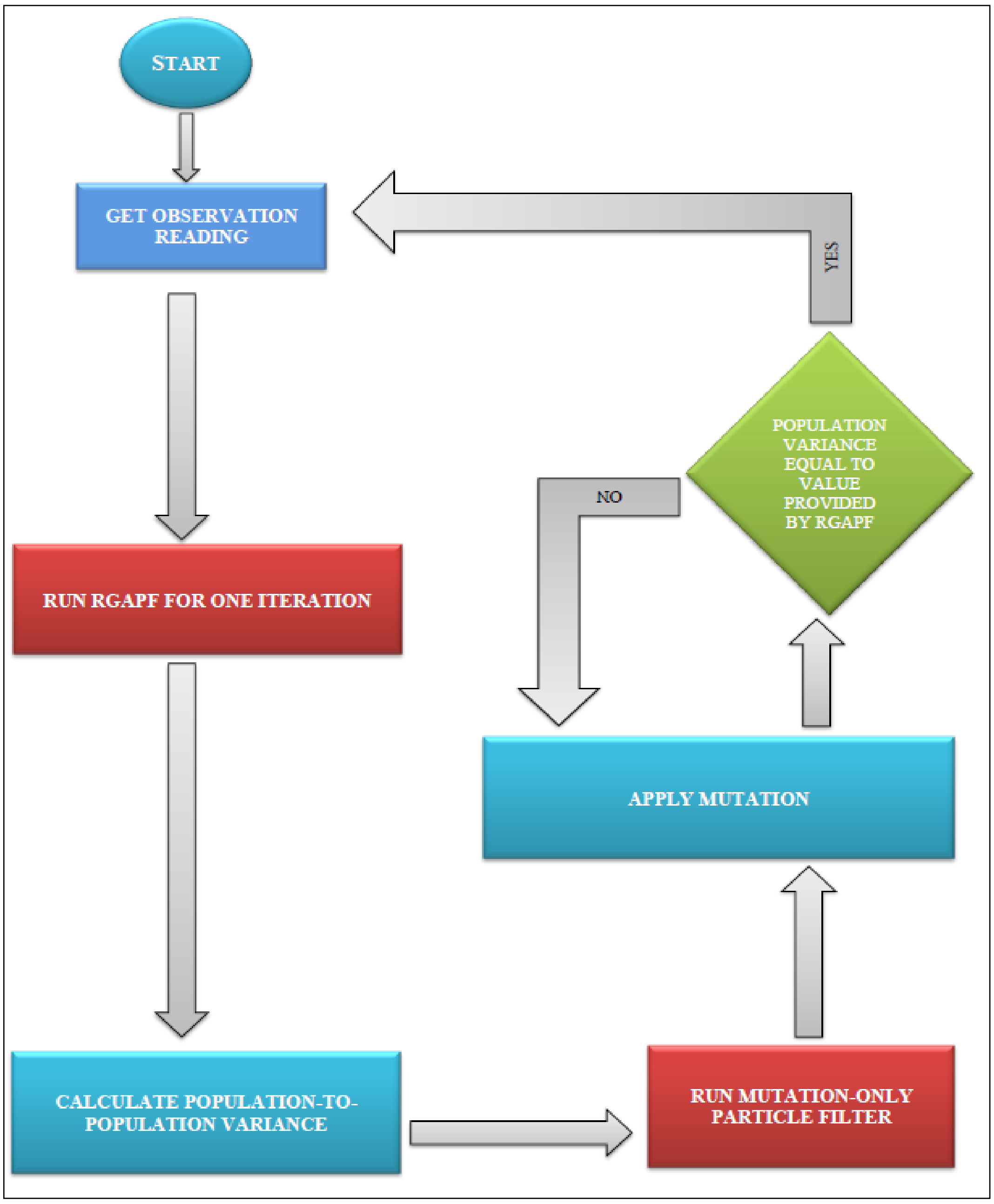

START
GET OBSERVATION READING
RUN RGAPF FOR ONE ITERATION
CALCULATE POPULATION-TO-POPULATION VARIANCE
RUN MUTATION-ONLY PARTICLE FILTER
APPLY MUTATION
POPULATION VARIANCE EQUAL TO VALUE PROVIDED BY RGAPF
NO
YES

Figure 6-2: Design of Experiment - Part 2 (Testing the Role of Recombination in Dimensional Scaling)

6.2.1 Expected Outcome of the Experiment

Before the results of this experiment are listed and analysed, it would be helpful at this stage to predict the behaviour of the experiment if our hypothesis is correct. In the light of our hypothesis, we can expect to make the following observations:

1. In an RGAPF, the selection operator will sample particles containing above average fitness building-blocks, which the recombination operator will then utilize by combining these building-blocks into single stings of above average fitness.
2. Due to the highly disruptive nature of the mutation operator, the performance of the mutation-only particle filter will deteriorate with an increase in the state-dimension and the number of particles.

The results of the experiments are presented next.

6.2.2 Results

Experiments were carried out using 50,100, 400 and 1000 particles. Figure 6.2 shows the comparison between the RGAPF and the mutation-only particle filter when the particle population was 1000. The red line represented the RGAPF performance while the blue line represents the performance of the mutation-only particle filter. The state-dimensions are on the horizontal axis. In figure 6.3, due to the high RMSE values of the mutation-only particle filter, the performance variation in the RGAPF cannot be observed. The performance of the RGAPF is hence shown separately in figure 6.4.

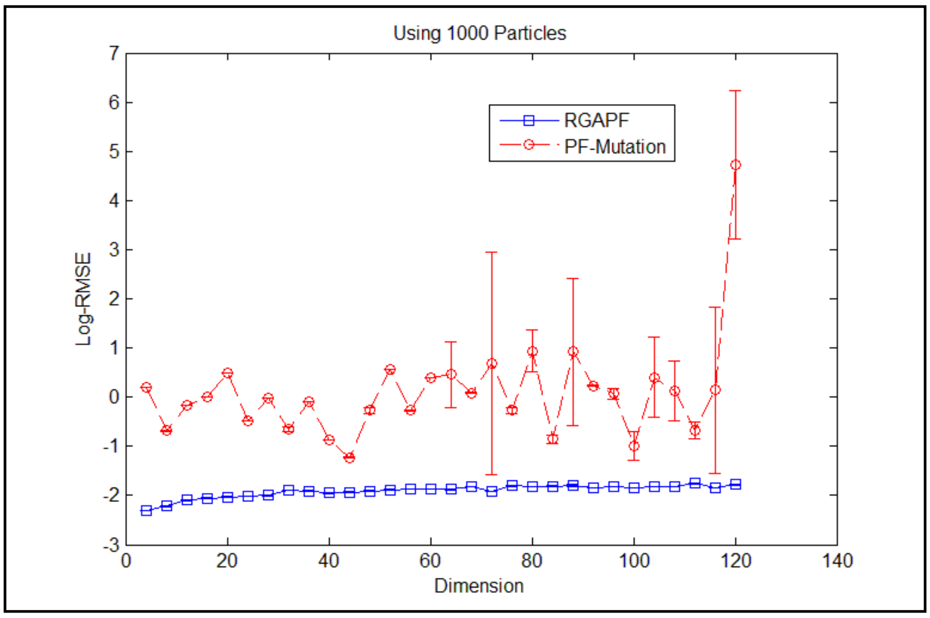


Figure 6-3: RGAPF vs. mutation-only Particle filter - Dimensional scaling (1000 particles)

The performance of the mutation-only particle filter shows the disruptive nature of the mutation operator. An increase in the state dimensions increases the change in population variance, and this increases its mutation rate. An increase in the mutation rate decreases the construction of higher-order hyperplanes and this significantly impacts the performance of the filter in higher-dimensions. The mutation-only particle filter also fails to perform in lower dimensions as an increase in the population size in lower dimensions also contributes to an increase in its mutation rate. A high mutation rate affects the performance severely, and this is reflected in the filter performance.

The performance of the RGAPF is shown in figure 6.4, where the different coloured lines represent RGAPFs with different number of particles. Unlike the mutation-only particle filter, the RGAPF performance appears to follow a regular trend. The performance slowly deteriorates as the state dimensions are increased, though the loss in performance is gradual. The graph also shows that there is an improvement in performance as the number of particles are increased, however observing the graphs closely, it can be noticed that the performance of the RGAPF with a 1000 particles is similar to the performance of the RGAPF with 400 particles, hence there seems to be a convergence in performance. The plausible explanation behind this behaviour has already been explained in the discussion at the end of last chapter. However it will briefly be mentioned here.

The rationale behind the success of RGAPF in higher-dimensions can be explained via the building-block hypothesis or the constructive ability of recombination, while the rationale behind a similar performance exhibited by RGAPFs with different particle population can be explained via the phenomenon of implicit-parallelism. Holland [Hol75] showed that a string of length l is an example of 2^l schemata and the population will usefully process $O(\mu^3)$ schemata. This result is known as implicit-parallelism and is quoted as one of the main reasons of the success of genetic algorithms [Mit98]. According to this phenomenon, while the algorithm may be explicitly sampling and evaluating a smaller number of parent particles, implicitly it is sampling from a much larger population set. Hence in the graph below, a similar performance is observed between particles with varying particle population size, and the improved performance is not significant when increasing the particle population.

The results of this experiment seem to validate the role of recombination in creating hyperplanes of higher order.

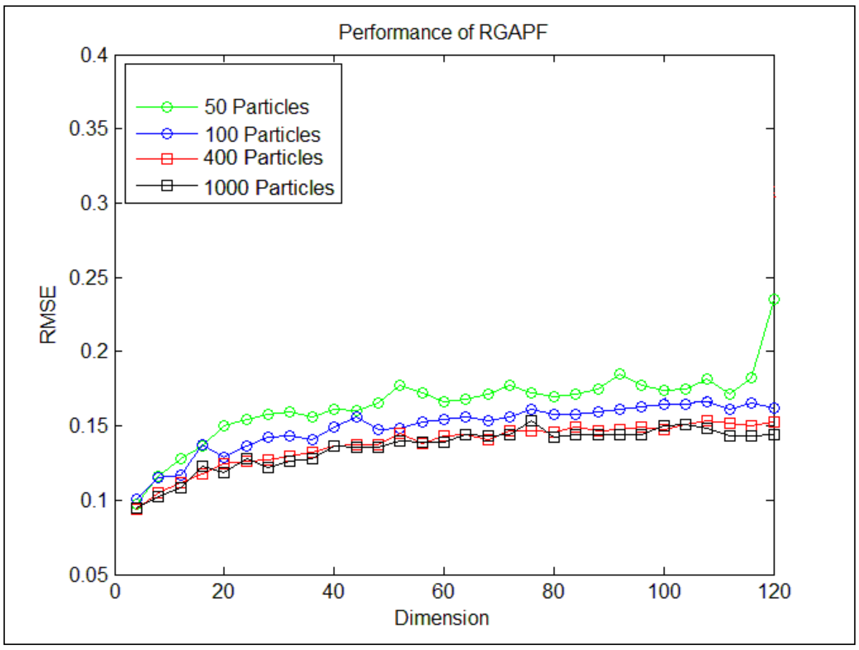


Figure 6-4: RGAPF Performance - Experiment 2

Error bars have not been included in the graphs above since the deviations were of the order of 0.01 or less. The next table shows the RMSE values for the RGAPF and the mutation-only particle filter when 1000 particles were used.

Table 6.1: RGAPF vs. Mutation-only Particle filter - Performance Comparison with 1000 particles

Dimension	RGAPF-RMSE	Particle Filter (Mutation only)-RMSE
4	0.097232	0.5897
8	0.105294	1.0049
12	0.106383	1.2495
16	0.116707	1.3560
20	0.126443	1.5520
24	0.122536	2.1761
28	0.118104	0.2769
32	0.127161	0.7816
36	0.131797	0.2589
40	0.135873	0.9517
44	0.136806	0.3825
48	0.128795	0.7073
52	0.137552	0.3605
56	0.140844	0.2235
60	0.139548	1.8048
64	0.148815	0.1784
68	0.141255	0.6980
72	0.141009	4.5460
76	0.149298	4.2187
80	0.150057	0.5626
84	0.148386	0.4161
88	0.142062	6.0226
92	0.142119	1.8932
96	0.148604	54.0948
100	0.166168	2.1573
104	0.142254	52045.0221
108	0.15092	2.9164
112	0.147308	1.9581
116	0.147458	315.5814
120	0.151877	5.0024

Table 6.1 shows a consistency in the performance of the RGAPF with increasing dimension. The performance of the mutation-only particle filter deteriorates with an increase in the state dimensions. Another observation that can be made after observing the RMSE values of the mutation-only particle filter is that even in low-dimensions; its performance is worse compared to the benchmark algorithms that were tested in the previous chapter. The results clearly show the superiority of the RGAPF over the mutation-only particle filter. The presence of a recombination operator in the RGAPF can thus be credited for its scalability. Thus the statement that recombination acts like a variable rate mutation can be refuted in a particle filtering setup. The performance of the two algorithms is significantly different (p-values << 0.001).

Graphs for the results obtained when the numbers of particles were 50, 100, 400 and 1000 are shown next.

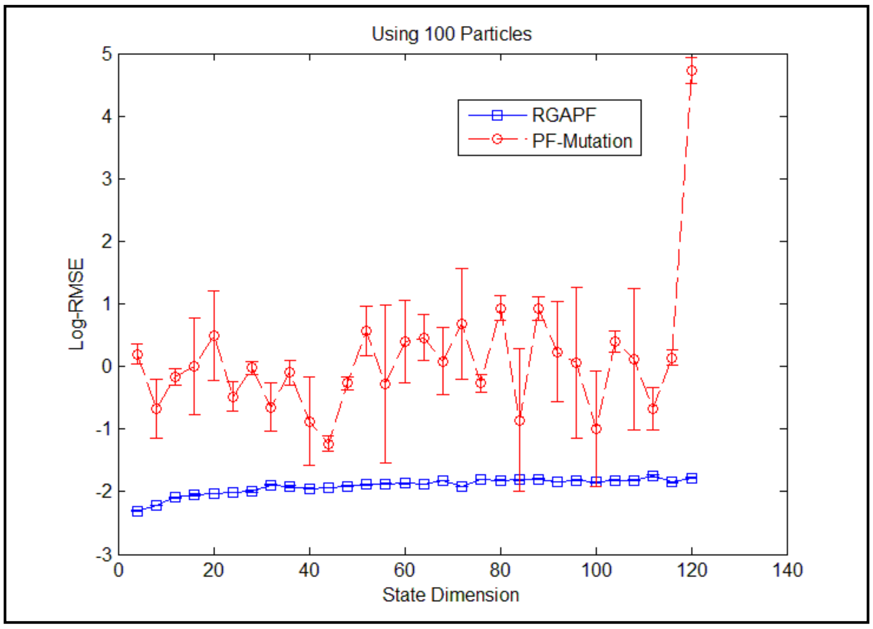


Figure 6-5: RGAPF vs. Mutation-Only Particle Filter - 100 Particles

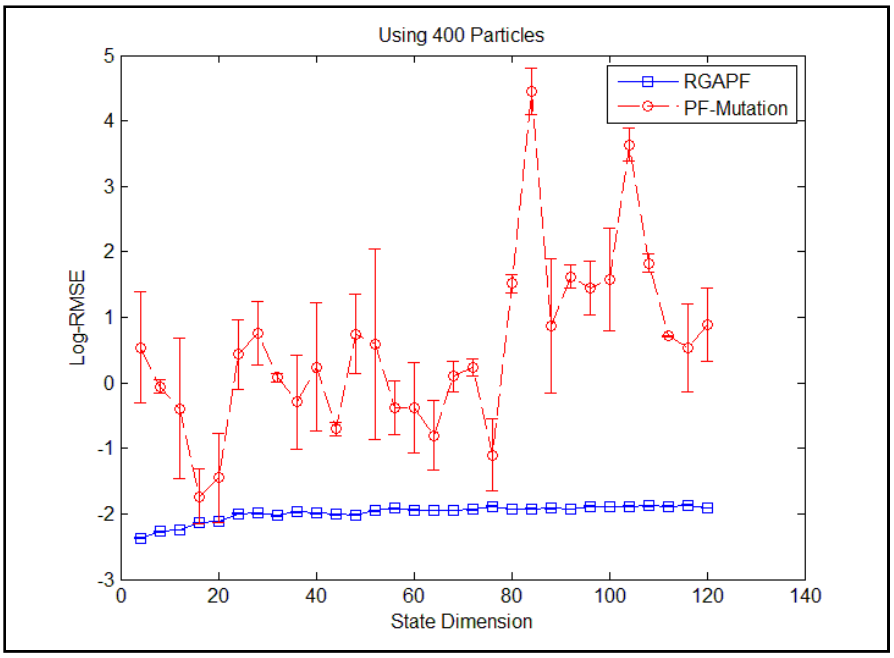


Figure 6-6: RGAPF vs. Mutation-Only Particle Filter - 400 Particles

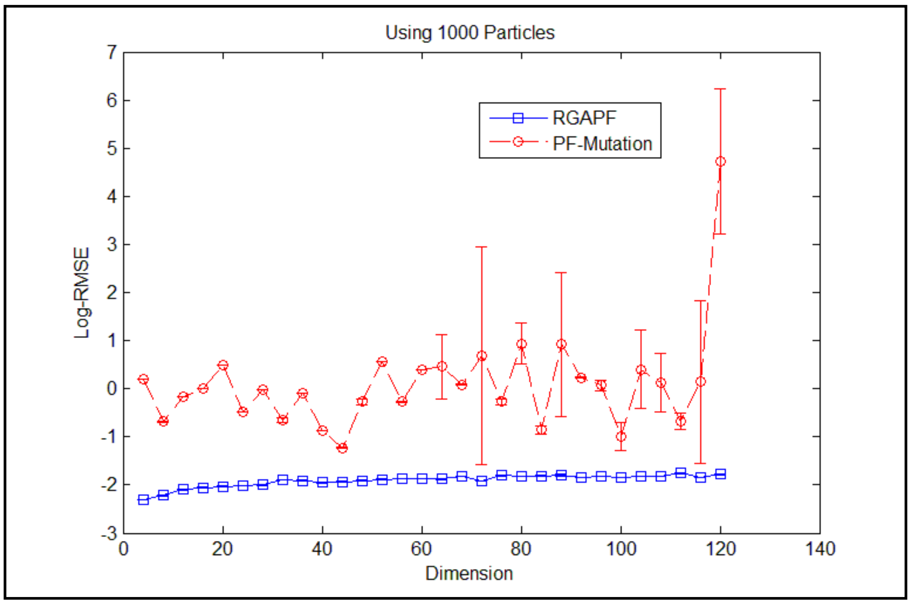


Figure 6-7: RGAPF vs. Mutation-Only Particle Filter - 1000 Particles

The inconsistent performance of the mutation only particle filter shows that as the state dimensions are increased, it is unable to combine lower-order hyperplanes to construct higher order hyperplanes of above average fitness. The results of the above experiment follow from the conclusions drawn by Spears in [Sp98], where he concluded that the mutation operator does not contribute to the constructive ability of a genetic algorithm.

6.2.3 Discussion

The results of the experiment follow directly from the building-block hypothesis. The ability of the recombination operator to combine building-blocks onto a single string translates to enabling it to scale to higher dimensions. The constructive property of the recombination operator also makes the algorithm to be adaptive and guides the search to be carried out more efficiently in the search space. The results also showed the inability of the mutation operator to mimic the performance of the recombination operator. An increase in the mutation rate made the algorithm performance to deteriorate, a phenomenon that can be explained by the highly disruptive nature of the mutation operator.

The approach followed in this thesis relies on the constructive property of the recombination operator. Based on the similarities between a genetic algorithm and a particle filter, GA theory was used to explain the working of a particle filter, and then using the building-block hypothesis, it was recommended that the addition of a recombination operator would introduce building-block like effects in a particle filter, and this would help address the phenomenon of sample impoverishment in higher-dimensions. The results of the experiment carried out in the previous chapter validated our approach, however in this chapter we tested whether the building-block like effects are due to recombination or can a similar performance be achieved by modifying the mutation operator. Observing the results of this experiment, we can arrive at the following conclusions:

- The recombination operator effectively combines components of the selected parents onto single particles such that the resultant particle has a greater weight compared to the parents.
- The creation of high weight particles is able to guide the search to be carried out more efficiently and around the expected value of the posterior density.
- Increasing the state dimensions does not have any significant effect on the performance as the algorithm is able to adapt accordingly.
- Mutation alone is unable to mimic the constructive ability of a recombination operator and increasing the mutation rate further deteriorates the performance of the filtering algorithm.

These observations follow directly from the building-block hypothesis. In [MFH92], Mitchel et al., made a similar attempt to test the building-block hypothesis by using a novel royal-roads function. They compared the performance of a simple GA with other algorithms, most notably the *Random Mutation Hill Climbing* (RMHC) algorithm. The RMHC algorithm is described below:

1. Choose a string at random. Call this string *best-evaluated.*
2. Apply mutation at random. If higher fitness is achieved, then set *best-evaluated* to the resulting string.
3. Go to step 2 until an optimum string has been found or until a maximum number of evaluations have been performed.
4. Return the current value of *best-evaluated*.

The results of their experiments were favourable to the RMHC algorithm, with the number of iterations required for convergence being 6179 compared to the 61,334 for the GA. This success of RMHC algorithm compared to a GA could have been critical for the explanation provided by the Schema theorem, however on closer observation the authors discovered the phenomenon that they called hitch-hiking. After the issues raised by the authors were addressed, they ran the GA again on the same problem and were able to get convergence in only 696 iterations. Similarly the experiment carried out here seems to validate the building-block hypothesis and seem to indicate that the recombination operator is solely responsible for constructing high-order hyperplanes.

The next section uses four different recombination operators to test which recombination operator in an RGAPF scenario gives the best performance within a RGAPF. These operators have already been discussed in chapter 2.

6.3 Experiment 3 - Using Different Recombination Operators in a RGAPF

The experiment performed in the previous section further strengthened our belief in the hypothesis of this thesis. In chapter 6 it was shown that the RGAPF was able to scale to higher dimensions compared to the benchmark algorithms, while the results of the previous section further confirmed that the recombination operator was responsible for the scalability to higher-dimensions.

An arithmetic recombination operator had been used for the first two experiments. Now a comparison will be carried out between the performances of the RGAPFs using three different recombination operators. The operators under investigation are:

1. The mean-centric recombination UNDX
2. The parent-centric recombination mPCX
3. N-Point recombination

The recombination and mutation rates are set equal the experiment carried out in chapter 5, however the parameter settings recommended by Ono et a., in [OK97] is used for UNDX, while the parameter setting for mPCX proposed by Deb et al., in [DJ02] have been used. For the UNDX and mPCX, the number of parents selected are 10% of the current population size. For the n-point recombination, the crossover points were determined by the following equation:

Crossover Points = (Dimension of the problem / 4) - 1

This ensures that an ideal scenario for n-point recombination is created as the state-dimensions are scaled. The results of this experiment show that the mean-centric recombination operator, UNDX, performs consistently better than the other recombination operators within the RGAPF, even though the conditions were made to ensure that the scenario was ideal for the n-point recombination operator.

6.3.1 Results

The particle scaling and dimensional scaling phases for this experiment are similar to the experiment carried out in chapter 6. In chapter 6 the comparison was between PLA, PF-LW and RGAPF, however in this experiment we are comparing four RGAPFs with different recombination operators. The RGAPF with arithmetic recombination is used as a benchmark here. Figures 6.9 and 6.10 summarize the performance of the four RGAPFs under varying dimensions and particle population. The recombination rate was equal to 0.5 and the mutation rate was set equal to 0.02.

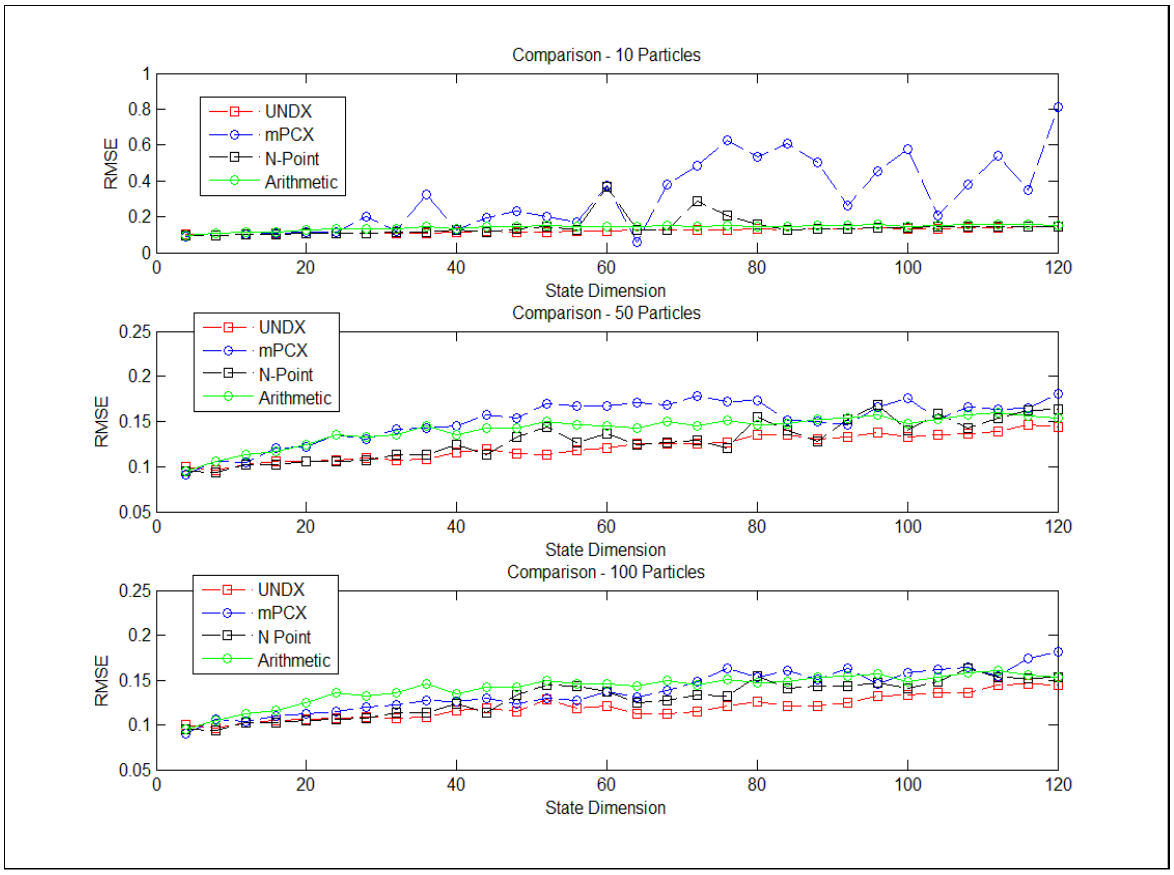


Figure 6-9: RGAPF with Different Recombination Operators – 10, 50 and 100 Particles

In figure 6.9, the performance comparison is carried out using 10, 50 and 100 particles. Observing the three graphs in this figure, the following observations can be made:

1. All the RGAPFs are able to scale to higher-dimensions.
2. The performance of the particle filters improves with the increase in the number of particles, though this improvement is not considerably large.
3. The RGAPF with mean-centric recombination (UNDX) appears to be the best performing particle filter.

When the number of particles was only 10, the arithmetic and mean-centric recombination filters performance were relatively similar; however in this scenario the RGAPF with parent-centric recombination had the worst performance. Increasing the number of particles from 10 to 50 to 100 improved its performance and brought its RMSE down. Similarly an increase in the number of particles improves the performance of UNDX compared to the arithmetic recombination operator.

The performance comparison when using 200, 500 and 1000 particles is shown in figure 7.10. The UNDX operator continues to be the best performing recombination operator amongst the other three operators. The performance of mPCX improves with the increase in number of particles, however by comparison it is still the worst performing recombination operator.

The N-Point recombination is not recommended for real-optimization [BD01, Tal09] and the results of this experiment show the inconsistent performance of this recombination operator in this setting.

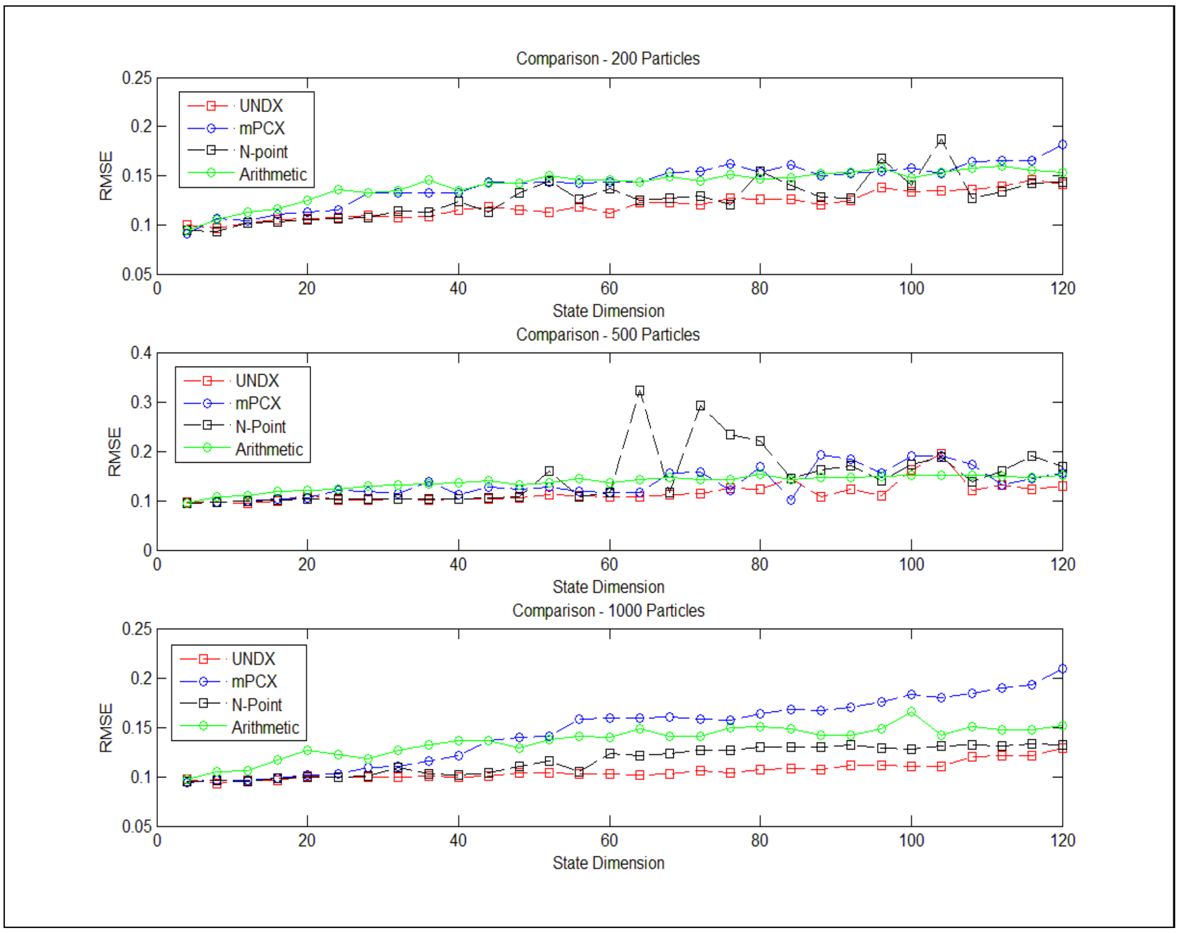


Figure 6-10: RGAPF with Different Recombination Operators – 200,500 and 1000 Particles

6.3.2 Discussion

So far in this thesis, it has been established that the addition of a recombination operator inside a particle filter would introduce building block like effects that will help enable the particle filter to scale to higher dimensions by addressing the issue of sample-impoverishment.

The comparison of the performance of the RGAPFs with different recombination operators was carried out in the previous sub-section. The real-recombination operators have already been discussed in chapter 2. The choice of the recombination operators was based on an earlier comparison carried out by Deb et al., in [DJA03]. All the recombination operators used were able to help scale the particle filter to higher-dimensions, however the UNDX operator stood out as the best performing recombination and provided the best estimates of the posterior for varying population size and state-dimensions. The UNDX operator, a mean-centric recombination operator, performs consistently better than the other three. In [DAJ02], Deb et al., carried out a comparison between real-recombination operators and showed the superiority of the mPCX recombination operator over other real-recombination operators. They used the UNDX operator to design their mPCX operator, however they ensured that apart from being parent-centric, their proposed operator was computationally less expensive than the UNDX. The similarities between the UNDX and the mPCX can be seen in the table 6.2.

Table 6-2: The mPCX and UNDX - A Comparison

	mPCX	UNDX
1	Select a population of parents.	Select a population of parents.
2	Choose one parent out of the population.	
3	Find a mean-parent vector.	Find a mean-parent vector.
4	Find a direction vector for each parent.	Find a direction vector for each parent.
5	Find the mean of perpendicular distance of each parent from the direction vector, D.	Find the mean of the perpendicular distance of each parent from the direction vector, D.
6	Add a fraction of D and the direction vector to the ***selected parent***.	Add a fraction of D and the direction vector to the ***mean-parent vector.***

The main difference between the above operators is that in the mPCX the resultant offspring is closer to the chosen parent, while in the UNDX, the resultant offspring is closer to the mean of the selected parents. In a particle filtering scenario the average of the particle population is required to be the expected value of the posterior density function. The UNDX operator carries out its search around the expected value and hence it is better able to search the space across different dimensions and hence perform better than any other operator. The mPCX operator however is biased towards only one parent and the offspring is created near the selected parent, hence it is only able to search the space closer to the chosen parent.

It can thus be concluded that a recombination operator, that carries out a search in the proximity of the expected value of the posterior will perform better than other operators in a particle filtering setup. A table listing the RMSE for the four RGAPFs when the number of particles was equal to 500 is given below. For a complete table please refer to the Appendix.

Table 6-3: RGAPF with different recombination operators - dimensional scaling (500 particles)

Particles	Dimensions	UNDX	mPCX	N-Point	Arithmetic
500	4	0.096054	0.095468	0.094351	0.095621
500	8	0.096054	0.095468	0.094351	0.107031
500	12	0.097333	0.096741	0.096229	0.108709
500	16	0.095458	0.099586	0.098249	0.118631
500	20	0.098237	0.102808	0.101021	0.120298
500	24	0.110884	0.105046	0.10222	0.123151
500	28	0.101241	0.120236	0.102267	0.128605
500	32	0.100901	0.119395	0.103772	0.132023
500	36	0.102229	0.11344	0.103554	0.134419
500	40	0.101481	0.138874	0.103248	0.13485
500	44	0.102807	0.111744	0.104141	0.139471
500	48	0.10402	0.127265	0.105011	0.131844
500	52	0.105407	0.122702	0.10686	0.13597
500	56	0.111761	0.1269806	0.1598872	0.143607
500	60	0.110134	0.118796	0.10815	0.135403
500	64	0.107065	0.117021	0.116305	0.141383
500	68	0.1066742	0.1168613	0.323123	0.146992
500	72	0.11214	0.1554895	0.116092	0.142712
500	76	0.115143	0.156794	0.291905	0.141933
500	80	0.124055	0.1202	0.234025	0.152737
500	84	0.1239028	0.1695646	0.219936	0.143107
500	88	0.145587	0.100337	0.1441017	0.146122
500	92	0.1076878	0.193196	0.1628527	0.147065
500	96	0.1238886	0.184593	0.1716893	0.148994
500	100	0.1089096	0.1549	0.141303	0.151121
500	104	0.1620856	0.1893469	0.172226	0.151813
500	108	0.1939816	0.1895	0.1890185	0.15127
500	112	0.1207081	0.1734332	0.1370336	0.148488

Particles	Dimensions	UNDX	mPCX	N-Point	Arithmetic
500	116	0.1324	0.13242	0.1607734	0.146051
500	120	0.12324918	0.144	0.19	0.150485

6.4 Summary

The discussion at the end of the last chapter used the Building-Block-Hypothesis to explain the success of the RGAPF in high-dimensional state-spaces. The Schema theorem explains the success of a genetic algorithm but focuses mainly on the destructive aspects of mutation and recombination, the Building-Block-Hypothesis however emphasizes the importance of recombination for creating higher-order higher-fitness schemata. Hence in GA theory, recombination is considered more important that the mutation operator. Holland himself added mutation as an insurance policy [Mit98] and the mutation rate is usually recommended to be around the order of 0.01. The Evolution Strategies literature however focuses mainly on the mutation operator, and Fogel et al., later described recombination as a variable rate mutation operator. The previous chapter may have shown the constructive property of a genetic algorithm, however the main focus in chapter was to determine whether the construction is mainly due to recombination or whether recombination is a variable rate mutation operator. The first experiment carried out in this chapter was to test the building-block hypothesis.

The results of this experiment show the collapse of the mutation only particle filter in high dimensions, while the results of the RGAPF are similar to the results obtained in the experiment carried out in the previous chapter which were explained using the Building-Block-Hypothesis. The comparison of the performance of the mutation only particle filter and the RGAPF highlights the importance of recombination in the adaptive behaviour of the GA. Comparing the RGAPF with different recombination operators we found that the RGAPF with a mean-centric recombination operator (UNDX) was able to outperform the RGAPFs with other recombination operators.

The results of the experiment showed the mPCX operator to be the worst performing in a particle filtering scenario compared to other operators. These results are completely different to the analysis carried out by Deb et al. in [DAJ02]. However Deb’s analysis was on an optimization problem with one candidate solution, while a particle filter is required to create a population of particles that better represent the posterior density, hence it was concluded that a recombination operator that carries out search in the proximity of the expected value of the posterior is more suited for this task.

Based on this conclusion, two recombination operators are proposed in the next chapter that are designed specifically for the RGAPF.

Chapter 7

Recombination for High-Dimensional Particle filtering

In the previous chapter it was shown that the UNDX operator consistently performed better than other recombination operators within an RGAPF. The results of the previous chapter led to the conclusion that the UNDX is able to search the space that is near the expected value of the posterior and hence is better able to guide the search as more observations become available. However the UNDX operator is computationally expensive to implement compared to other recombination operators. Based on the conclusions of the previous chapter we propose a recombination operator in this chapter, the *mean-centric Gaussian recombination* (MCGR), for high-dimensional particle filtering that is based on the UNDX but with a complexity similar to the mPCX operator.

The performance of the proposed MCGR operator is compared with the UNDX and the arithmetic recombination operators on both simulated data and real end of day price data taken from the *London Stock Exchange* (LSE). The predictions provided by the MCGR is similar to the estimates provided by the UNDX.

This chapter can be divided into the following main sections:

- Section 7.1 recounts the results and conclusions of the previous chapter, and lists a few issues of the UNDX that have been noted in literature.
- We propose our recombination operator, the *mean-centric Gaussian recombination* (MCGR), in section 7.2.

- In section 7.3 a comparison is carried out between MCGR, the UNDX and the arithmetic recombination within an RGAPF using simulated data.
- In section 7.4, a final comparison is carried out between the benchmark algorithms, the proposed RGAPF and a hybrid CMA-ES particle filter taken from [SH12a].
- This chapter concludes with a summary in section 7.5.

7.1 Approach

The experiments carried out so far in this thesis confirm that the addition of a recombination operator is able to address the sample-impoverishment phenomenon encountered by particle filters in higher dimensions. The last experiment carried out in the previous chapter showed that the mean-centric recombination operator, UNDX, when used within an RGAPF, is able to provide the best possible estimates of the posterior density function. The *uni-modal normal distribution crossover operator* (UNDX) was proposed by Ono et al., in [OK97]. This mean-centric recombination assigns more probability to the creation of offspring near the mean of the selected parents. It was concluded in the previous chapter that since the created offspring are near the expected value of the posterior density, the UNDX operator is better able to guide the search with each arriving observation. However in [DJA03], Deb et al., while proposing their mPCX operator, noted that the UNDX is computationally expensive to implement. In this and the next we will discuss the workings of a UNDX operator and propose a modified version of UNDX which is not computationally expensive to implement.

Consider a population of particles as shown in figure 7.1. A particle represents a sample that is drawn from the posterior distribution. For a multi-dimensional density function, each particle is a vector where each component represents a particular dimension of the density function. Hence a particle may be made up of components that have a high plausibility of being sampled in that dimension from the density function and components that have a low plausibility. In figure 7.1, a particle population is shown that represent a 9-dimensional density function. The colour intensity in each particle shows components that have a high probability of being sampled. Lower colour intensity shows a lower probability of being sampled.

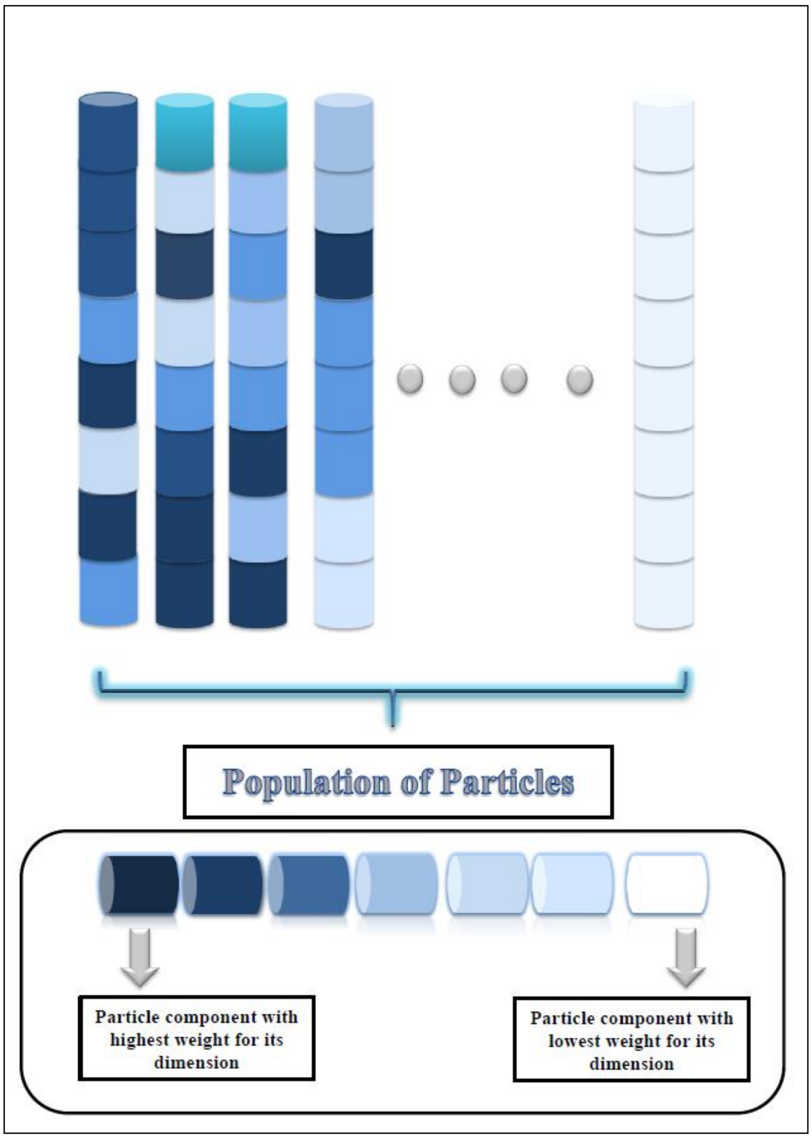

Population of Particles
Particle component with highest weight for its dimension
Particle component with lowest weight for its dimension

Figure 7-1: A high-dimensional particle population

In an RGAPF after the weight update these particles will undergo selection, recombination and mutation. In case of an arithmetic recombination operator, the selected parents will be combined based on the following equation:

$$Offspring = 0.7\ (Parent\ 1) + 0.3\ (Parent\ 2)$$

Applying recombination at each iteration would have ensured that the particle population is updated continuously. In chapter 6, a comparison of four different recombination operators within the RGAPF showed that the best possible estimates were achieved when the UNDX operator was used. In a UNDX operator, a greater probability is assigned to the creation of an offspring near the mean of the $(\mu - 1\ )$ selected parents.

Consider the equation of the UNDX operator:

$$y = g + \sum_{i=1}^{\mu-1} w_i \left|d^i\right| e^i + \sum_{i=\mu}^{n} v_i D e^i \tag{7.1}$$

Where w_i and v_i are standard zero-mean normally distributed variables, and n is the total number of individuals in the population. The steps involved in carrying out the UNDX recombination and the definition of the terms in equation 7.1 are as follows:

1. *(μ - 1)* parents x^i are randomly selected from the population.
2. The mean value g of the selected individuals is this computed.
3. Then, *(μ - 1)* direction vectors, $d^i = x^i - g$ are generated. The variable e^i, denotes direction cosines $d^i/\left|d^i\right|$.
4. Given a randomly selected individual x^μ, the length D of the vector $(x^\mu - g)$ orthogonal to all $e^i$is calculated.
5. An offspring y is created using equation 7.1.

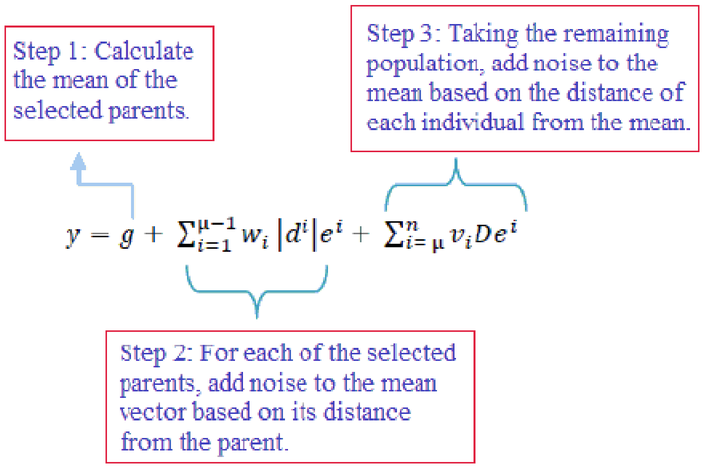
The offspring created would thus be located within the vicinity of the mean of the selected parent population. Since the selection operator has a high propensity to select parent particles that have a higher weight, hence the UNDX operator would be able to create an offspring that lies within the expected value of the density function. This may be a plausible explanation for its superior performance within a particle filtering setup compared to other recombination operators.

In [DJA03], Deb et al., proposed the parent-centric recombination mPCX which they based on the UNDX. They compared its performance with the UNDX on three different test problems; the ellipsoidal function, the Schwefel's function and the Rosenbrock's function. The mPCX outperformed the UNDX on all these tests, however the Deb et al., had used a *minimal generation gap* model (MGG) and the three test problems are optimization problems, unlike the filtering problem where the objective is to find the best possible population to represent the posterior density. However while proposing the mPCX, Deb et al., noted that the computational complexity of the UNDX is $O(\mu^2)$, compared to $O(\mu)$ for the mPCX. An increase in the population size would drastically affect the computational speed of the UNDX. Looking at equation 7.1, we can observe that the UNDX uses the selected parents and then the whole population to adjust the position of the offspring. The mPCX operator, given in equation 3.10, however only uses the selected parents. This brings the complexity of the mPCX down to the order of $O(\mu)$.

In the next section we propose the *mean-centric Gaussian recombination* operator (MCGR), that is based on the UNDX, however we modify it to bring it's computationally complexity down to $O(\mu)$.

7.2 The Mean-Centric Gaussian Recombination

The calculations involved in a UNDX operator can be divided into 3 main steps. The first two steps use the selected parent population, while the third step uses the whole population of individuals.


Step 1: Calculate the mean of the selected parents.
Step 3: Taking the remaining population, add noise to the mean based on the distance of each individual from the mean.
$y = g + \sum_{i=1}^{\mu-1} w_i \left|d^i\right| e^i + \sum_{i=\mu}^{n} v_i D e^i$
Step 2: For each of the selected parents, add noise to the mean vector based on its distance from the parent.

The second and third steps use the distance of the selected individual from the mean-vector to add a small Gaussian noise. The amount of zero-mean Gaussian noise is given by w_i and v_i in the above equation. It was proposed by Kita et al., in [KY99], that the variance of w_i should be equal to $1/\sqrt{\mu - 2}$, while the variance of v_i should be equal to $0.35/\sqrt{n - \mu - 2}$,

We propose that the third step in the calculation of the UNDX be omitted and to compensate for the perturbations that are added in this step, we set w_i to have a variance of $1/\sqrt{\mu}$. The *mean-centric Gaussian recombination* operator (MCGR) is given below:

$$offspring = g + \sum_{t=1}^{\mu-1} \left(\left(g - \overrightarrow{P_t} \right) \times \varepsilon_t \right)$$

Here $\boldsymbol{g}$ is the mean of the population of selected particles, $\overrightarrow{P_t}$ is the tth parent and ε_t is a zero-mean Gaussian noise with variance $1/\sqrt{\mu}$. This recombination operator is shown in figure 7.2. The colour of each component represents its plausibility of being sampled form the density function. The colouring scheme used is similar to the ones used in figures 7.1.

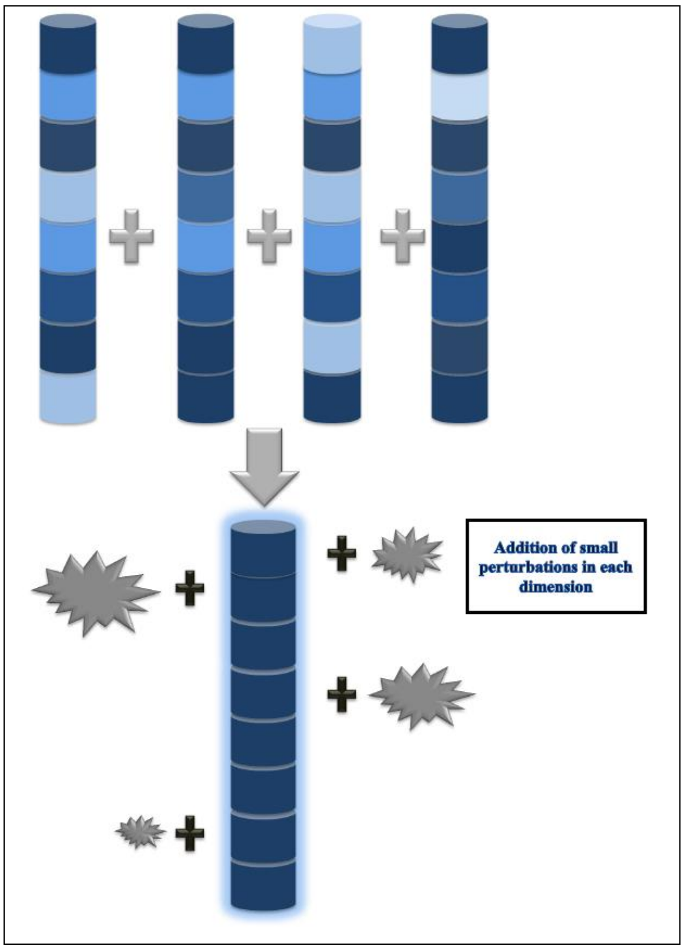


Figure 7-2: Operator 1 for High Dimensional Particle filtering

. Figure 8.3 shows that a mean-vector is calculated from the selected parents and then Gaussian noise is added to the mean-vector to obtain the offspring. The complexity of the MCGR compared to the UNDX has thus been brought down. However the performance comparison of the RGAPF with MCGR and the RGAPF with UNDX is carried out in the next section.

7.3 Experiment 4 – Performance of MCGR in RGAPF

The experimental set is similar to the experiment carried out in the last chapter where all the different recombination operators were compared, however only three RGAPFs are used here. The arithmetic recombination operator has been kept as a benchmark since it has been used in all the initial experiments.

7.3.1 Results

The performance of the MCGR within an RGAPF is similar to the performance achieved when the UNDX operator was used. We initially started with 20 particles and gradually increased the population size to 5000. As can be seen in figures 8.4 – 8.8, the performance of the UNDX and MCGR within the RGAPF was similar under different state-dimensions and particle population however by design the MCGR is computationally less expensive.

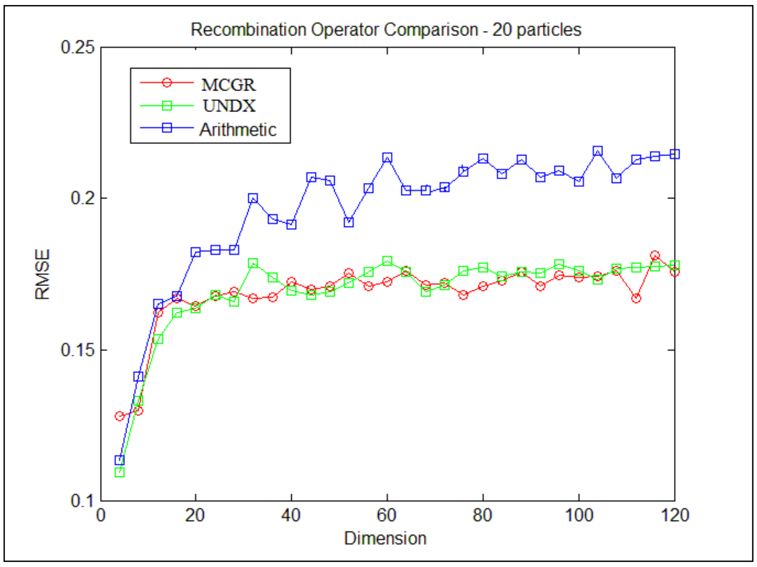


Figure 7-2: Proposed Recombination Operator - Performance Comparison (Dimensional Scaling with 20 Particles)

In figure 7.4, the performance of the three recombination operators within an RGAPF is shown when the population size was 20. Initially the performance of all the three filters was similar, however with an increase in the state-dimensions, the filters with UNDX and MCGR performed better compared to the filter with arithmetic recombination. As can be seen in figures 7.5-7.8, increasing the population size significantly improves the performance of the filters with UNDX and MCGR. However the UNDX operator uses the whole population of particles to generate an offspring while the MCGX uses only the selected parents. Even though both the operators have similar RMSE values (p-values > 0.05), the UNDX operator is a lot slower, a point noted by Deb et al., in [DJA03].

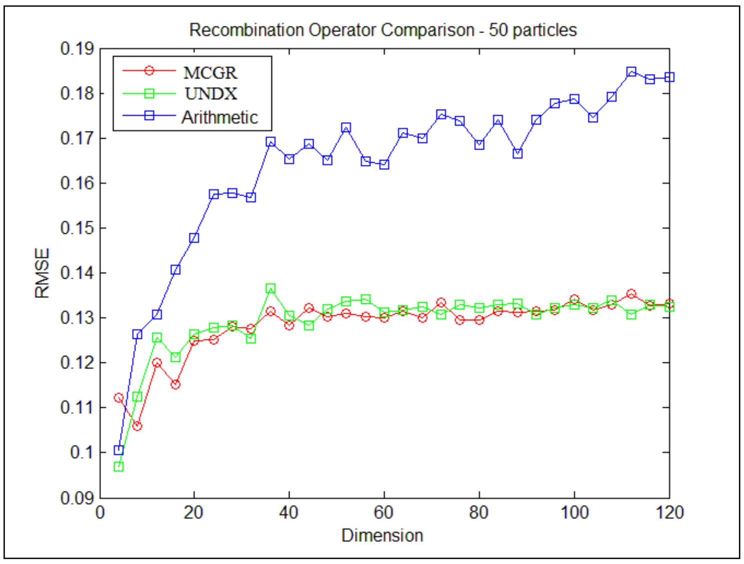


Figure 7-3: Proposed Recombination Operators - Performance Comparison (Dimensional Scaling with 50 particles)

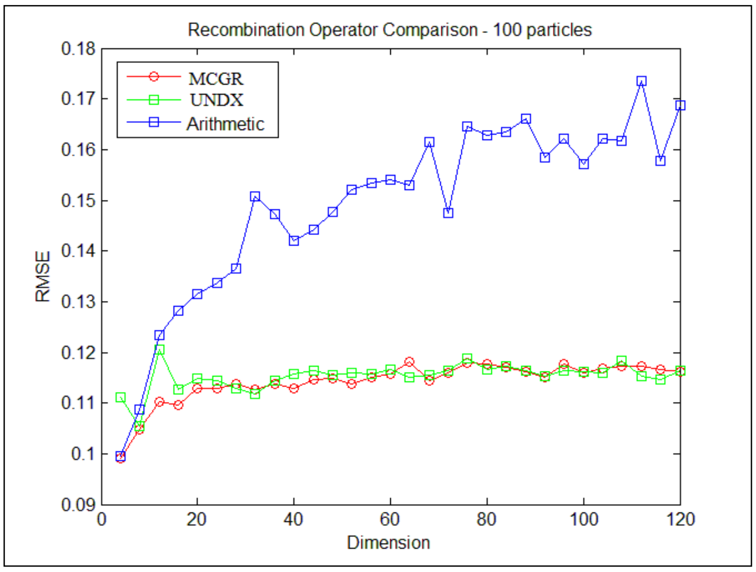


Figure 7-4: Proposed Recombination Operators - Performance Comparison (Dimensional Scaling with 100 Particles)

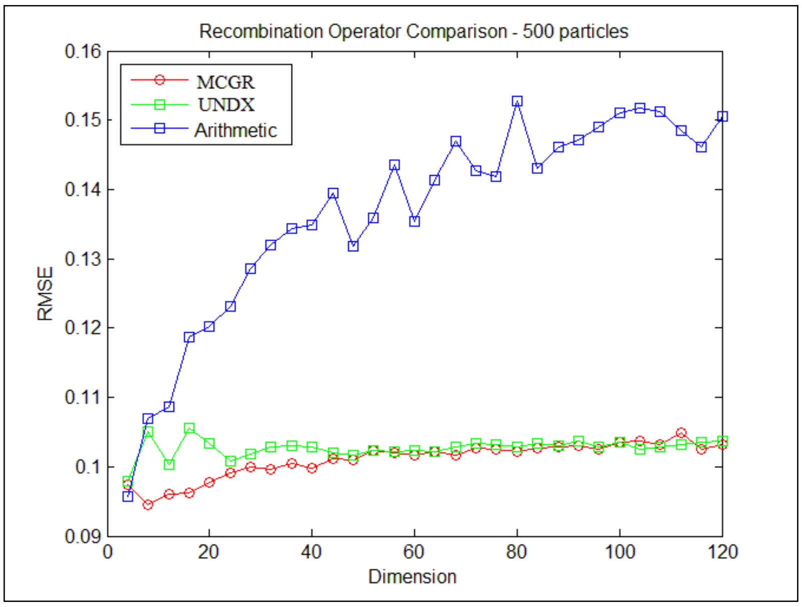


Figure 7-5: Proposed Recombination Operators - Performance Comparison (Dimensional Scaling with 500 Particles)

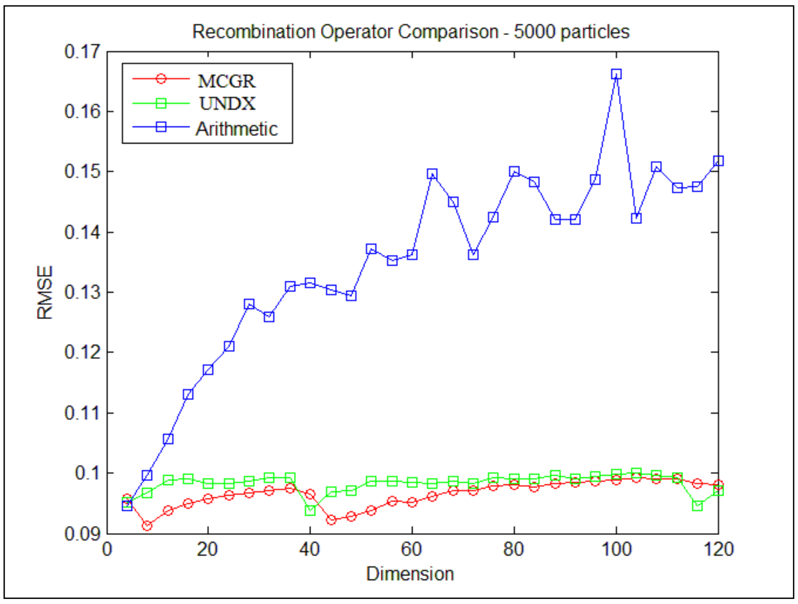


Figure 7-6: Proposed Recombination Operators - Performance Comparison (Dimensional Scaling with 5000 Particles)

7.3.2 Conclusions

In the previous section, the mean-centric Gaussian recombination operator was proposed and shown to have a performance similar to the UNDX operator in a filtering setup. The experiments carried out in the last chapter showed that compared to other recombination operators the UNDX operator was more accurate in predicting the posterior density under variable state and population size. It was concluded that the reason behind its superior performance is that it creates offspring near the expected value of the density and hence is better able to guide the direction of the search space. However compared to other recombination operators, the UNDX is computationally expensive to implement.

We analysed the UNDX operator and concluded that the complexity of this mean-centric recombination operator can be brought down by only using the selected parents and adjusting the variance of the added zero-mean Gaussian noise. The performance comparison carried out in 7.3.1 shows that our proposed operator had a similar prediction accuracy compared to the UNDX, and being of less complexity is a better operator to be implemented within an RGAPF.

In the next section we carry out a comparison similar to the one carried out in this section, but the observation series will not be simulated. We will be using real price data obtained from the London Stock Exchange. The FTSE-100 index and the time series of the assets that make up the FTSE-100 index will be used. The RMSE values are calculated using the estimates obtained when the pricing model was calibrated using MCMC methods.

7.4 Experiment 5 – RGAPF performance on FTSE-100 Time-Series

The observation series used in the experiments carried out so far in the thesis were simulated using equations 4.6-4.7. In this section real end of day price time series will be used. The time series is taken from the London stock exchange and provides the prices of the hundred assets that make up the FTSE-100 index from 4th January 2012 to 23rd November 2012. Along with the algorithms compared in the previous section, a hybrid CMA-ES particle filter [SH12] and the two benchmark algorithms, PF-LW and PLA, are also tested.

Before running the filtering algorithms we calibrate the stochastic volatility model given in equation 4.6 and 4.7 using MCMC techniques. Carrying out a complete MCMC analysis we were able to find the parameter values of the pricing models and the estimates of the stochastic volatility. These were then used as a benchmark to compare the performance of the filtering algorithms.

7.4.1 Experimental Setup

The experimental design in similar to the last chapter. The only difference is that instead of generating the time series, we use real time-series data when we wish to increase the dimension of the state. This way, we can scale up to 404 dimensions. (The FTSE-100 index is made up of 100 stock assets and the addition of the index itself will bring the state dimension to 404).

7.4.2 Results

As in the previous chapter, we employ plots to observe the results of our experiments. The benchmark algorithms were unable to perform in high-dimension, and provided significantly different results compared to the other tested algorithms (p-value < 0.001). The collapse of these algorithms in high-dimensions was expected as had been noted in [LW01] and [RB06]. Their collapse in higher-dimensions has also been shown experimentally in chapter 5 of this thesis.

The results on real end of day data are similar to the results obtained in the previous chapter, i.e., scalability to higher dimensions and the requirement of a less number of particles. Consider the diagram below, with only 50 particles:

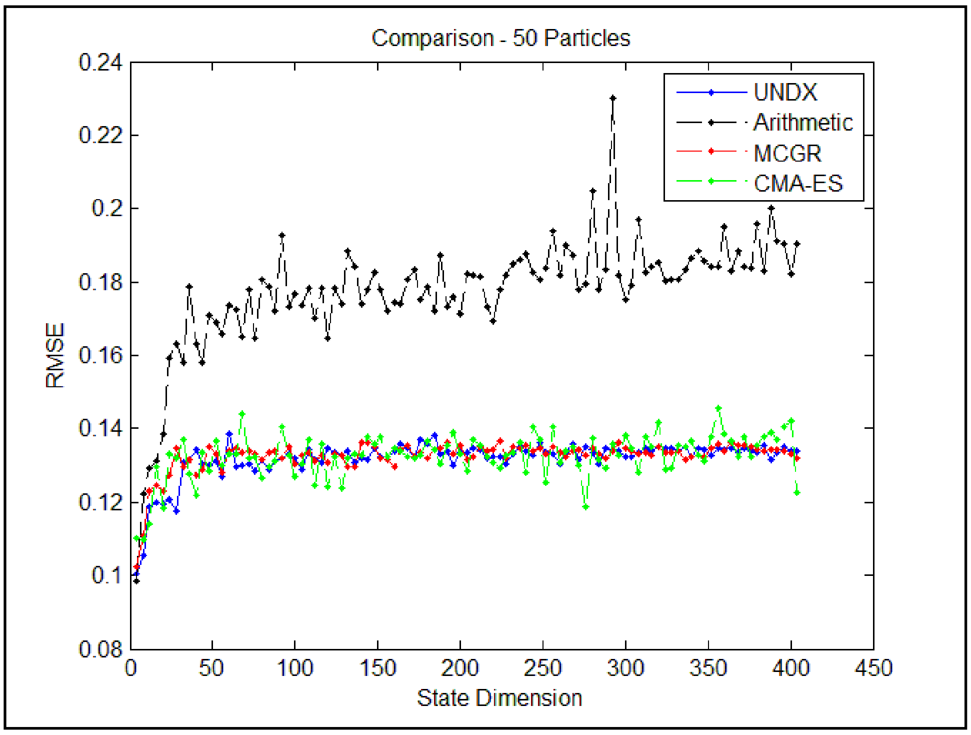


Figure 7-7: RGAPF Performance Comparison on Real Data- Dimensional Scaling (50 Particles)

With only 50 particles, the performance, i.e., RMSE of UNDX and MCGR is only of the order of 0.135, and increasing the state dimension has no effect on their. A similar result will be observed in the next few graphs. The CMA-ES particle filter also provides similar results (p-values > 0.05) and scales to higher-dimensions. The results of the benchmark particle filters have not been added to the graph as they were significantly different (p-values << 0.001) and collapsed after a few iterations even in a four dimensional scenario as stated by Liu and West in [LW01].

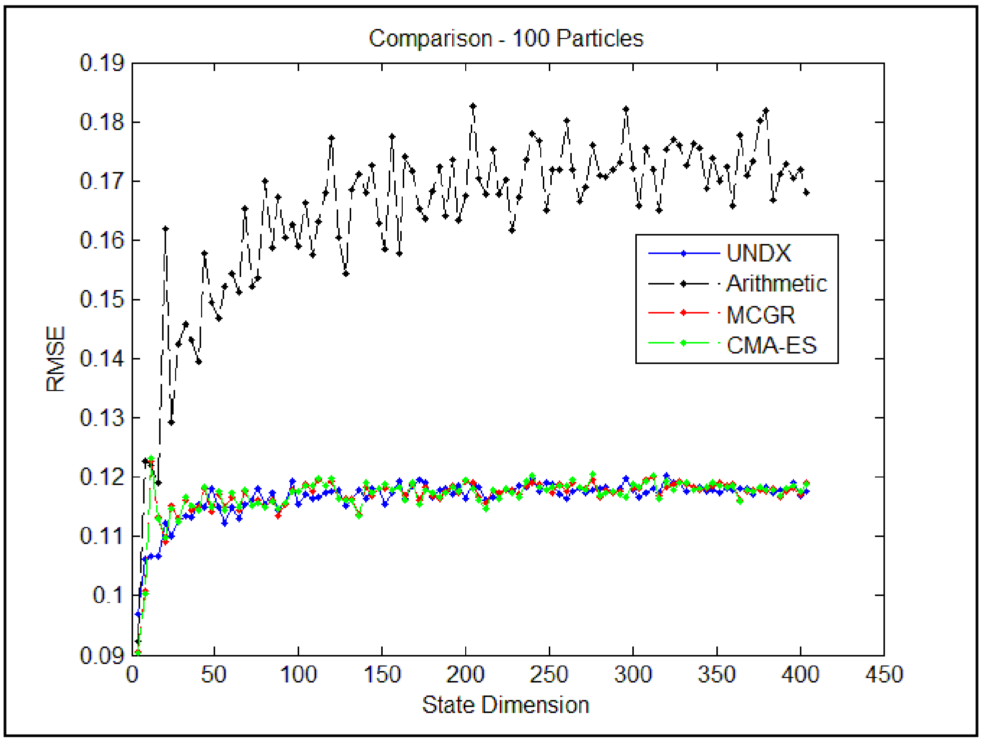


Figure 7-8: RGAPF Performance Comparison on Real Data- Dimensional Scaling (100 Particles)

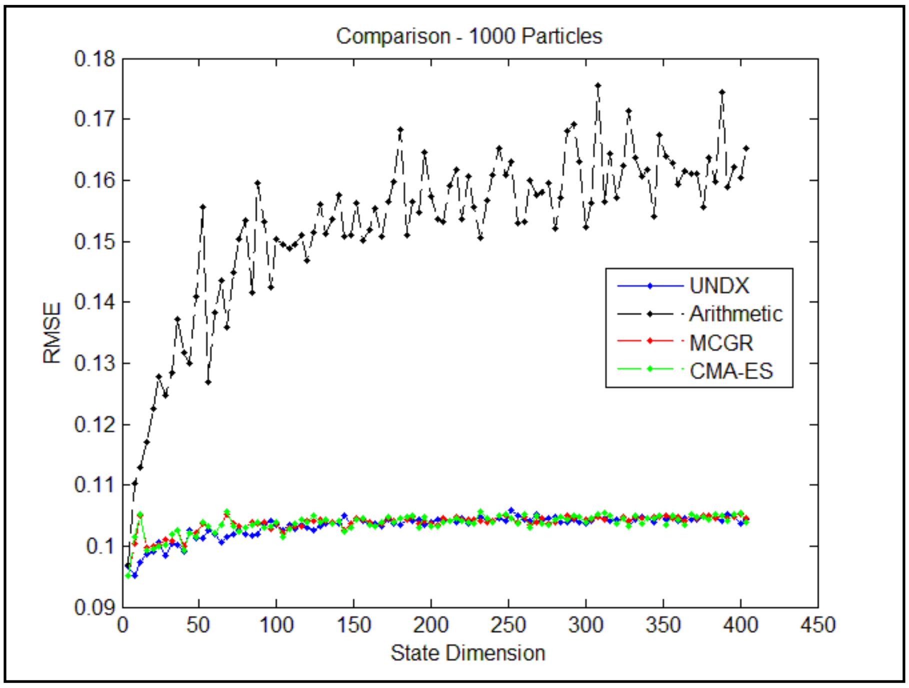


Figure 7-9: RGAPF Performance Comparison on Real Data- Dimensional Scaling (500 Particles)

A careful observation of the series of graphs show that an RMSE of the order of 0.1 is expected even when the state dimensions increases to as high as 400. The performance improvement using RGAPF with operators UNDX or MCGR is similar to the performance of the CMA-ES particle.

A table of results when the number of particles was 500 is given below. The p-values for the ANOVA test for the 4 algorithms is less than 0.0001 for the dimensions tested, as is clearly evident from the box-plot shown below.

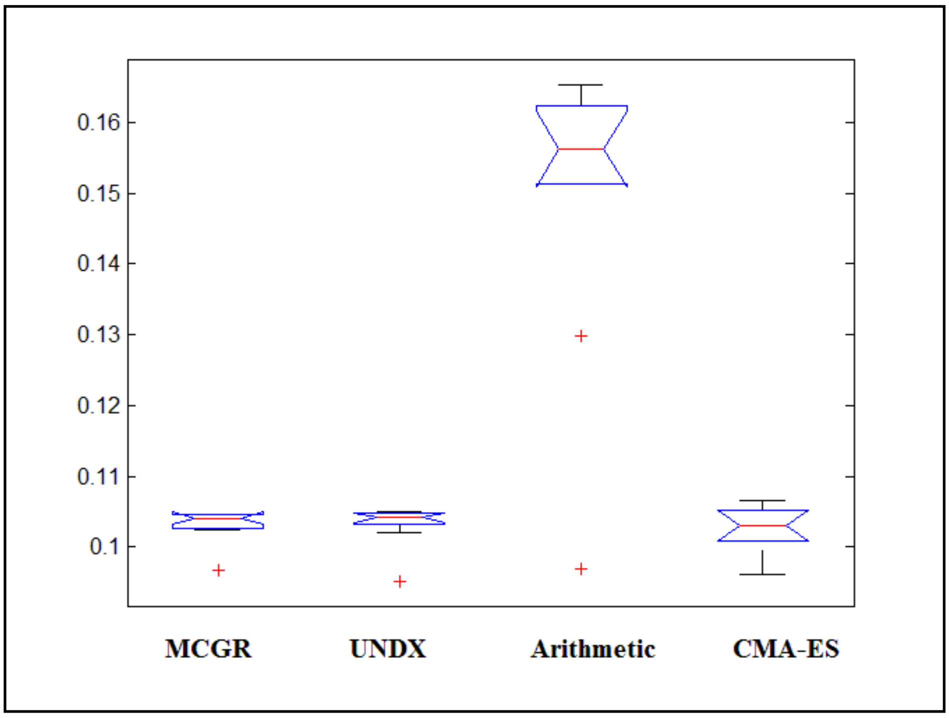


Figure 7-10: Box Plot – Performance Comparison (404-Dimensions)

Table7-1: RGAPF Performance Comparison - Real data (500 Particles)

Particles	Dimensions	MCGR - RMSE	UNDX - RMSE	CMA-ES RMSE
500	4	0.096675	0.09518	0.0961
500	44	0.102693	0.102088	0.1009
500	124	0.102493	0.104213	0.1031
500	164	0.103785	0.103274	0.1025
500	204	0.104335	0.103551	0.1006
500	244	0.104513	0.105019	0.1065

Particles	**Dimensions**	**MCGR - RMSE**	**UNDX - RMSE**	**CMA-ES RMSE**
500	284	0.103802	0.104744	0.1051
500	324	0.104644	0.104711	0.1040
500	364	0.104529	0.104049	0.1054
500	404	0.10428	0.104641	0.1029

When the results of the three algorithms listed above are analysed using the ANOVA method, the p-values show that they are similar performing algorithms (p-value > 0.5) as is evident in the figure below:

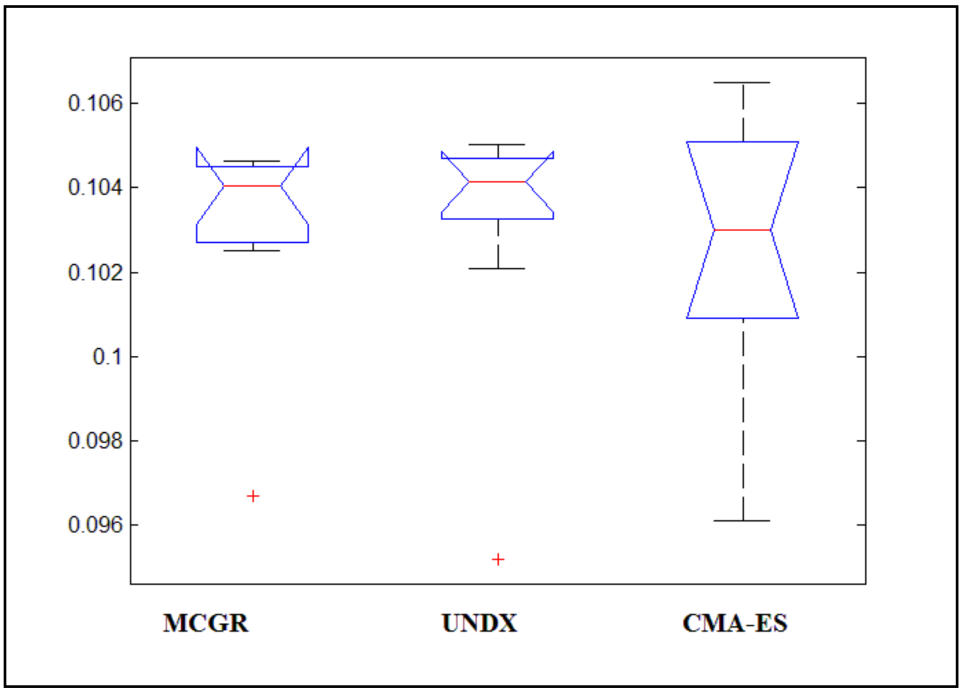


Figure 7-11: Box Plot: Performance Comparison between MCGR, UNDX and CMA-ES (404-Dimensions)

7.4.3 Discussion

The results of the performance comparison are similar to the results obtained when a simulated time series was used. The rationale behind the comparison carried out in this chapter was to check whether the results obtained in the previous experiments are applicable to real world scenarios. Hence it can be concluded that the approach and hypothesis of this thesis is applicable on real time series. A similar performance is achieved when using a hybrid CMA-ES particle filter; although, as can be seen in figure 7-11, it has more variance compared to the

RGAPF, and like the UNDX operator, it also has quadratic complexity. The proposed MCGR however has linear time complexity.

We can end our series of experiments with the following conclusions:

- GA theoretic arguments can be used to address issues in particle filtering algorithms.
- The addition of a GA layer in a particle filter is able to address the sample impoverishment issue in high-dimensions.
- Mean-centric recombination operators outperform other recombination operators in a particle filtering setup.
- The results of our experiments are valid on real stock price data.
- Further research is required to analyse particle filters using ES theory, since the hybrid CMA-ES particle filter provided results similar to the RGAPF on real-data. The complexity of the CMA-ES algorithm is quadratic however, hence and the RGAPF using MCGR being of linear-complexity provides the best possible result in the least amount of time of all the algorithms tested in this thesis.

7.5 Summary

The experiments of chapter 5 and chapter 6 further strengthened our belief in the hypothesis of this thesis that the addition of recombination in a particle filter will be able to address sample impoverishment in high-dimensions by introducing building-block like effects. Chapter 6 concluded with a comparison of different recombination operators within the RGAPF. The results of the comparison showed that the UNDX operator provided the best estimates of the posterior compared to other real-recombination operators. It was concluded that this is because the selection operator is able to select parents of above average weights and the UNDX then creates offspring near the expected value of the posterior. The addition of such offspring in the particle population is able to guides the algorithm to search the space more efficiently compared to other recombination operators. However the UNDX operator has a complexity that is far greater than other recombination operators.

In this chapter we proposed a recombination operator, the mean-centric Gaussian recombination (MCGR), that has a complexity equal to the mPCX operator while having a performance similar to the UNDX. Using both simulated data and real LSE data, it was shown that MCGR when used within the RGAPF provides the best performance compared to other recombination operators.

The final chapter of this thesis follows next, where the conclusion and future direction of research are discussed in detail.

Chapter 8

Discussion and Future Direction of Research

The objective of this thesis was to address the ensemble collapse observed in particle filters when the state dimensions are increased. The approach used in this thesis was based on exploiting the similarities between particle filters and genetic algorithms and then using genetic algorithm theoretic arguments to address the issues found in particle filters.

The results of our experiments show that we were successful in addressing the issues faced by particle filters in high-dimensional spatial systems, a phenomenon that has mathematically been shown to require an exponential number of particles for an accurate estimate. The experiments and results shown in this thesis open up new avenues of theoretical and practical investigation that marries recombinative genetic algorithm theory with Bayesian estimation theory. Further it can be concluded that an analysis of particle filtering methods by analysing them using genetic algorithm theory may provide another perspective to analyse their workings and performance.

8.1 Discussion of Results

Analysing the particle filter using GA theory led us to the conclusion that a generic particle filter with resampling and regularization is similar to a GA with selection and mutation. The missing element is a recombination operator. The missing recombination operator and its effect on the working of a GA were explained by revisiting the qualitative explanation of the working of a GA:

*"The simple GA increases the number of instances of low-order; short-defining length, high−observed−fitness schemas via the multi−armed−bandit strategy, and these schemas serve as building-blocks **that are combined, via recombination, into candidate solutions with increasingly higher-order and higher-observed fitness**."*

Translated in particle filtering terminology; a particle filter without recombination would be unable to combine the vector components of the particles, that represent the posterior correctly in a particular dimension, on a single string. This will result in particle degeneracy as the state-dimensions are increased.

Our hypothesis for this thesis was hence:

"The addition of a GA layer in a particle filter will increase the number of instances of low-order; short-defining length, high-observed weight particle components via the multi-armed-bandit strategy, and these particle components serve as building-blocks that are combined by recombination into candidate solutions with increasingly higher-order and higher-weights. Hence enabling the particle filter to scale to higher-dimensions"

A total of five experiments were carried out in this thesis to test this hypothesis.

8.1.1 Experiment 1

The first experiment was carried out to test the scalability of the proposed RGAPF to higher-dimensions. We had mentioned in our approach in chapter 3 that the addition of a recombination operator in a particle filter and lowering of the mutation rate would be able to address the issues that were faced by particle filters in high-dimensions.

We carried out our test on an SV estimation problem. Two benchmark algorithms; the PF-LW and the PLA were used. These algorithms are modified versions of a generic particle filter that were modified specifically for the SV estimation of common stocks. In our experiments we first tested the effect of increasing the number of particles on the effectiveness of these three algorithms. The particle filter performance is directly proportional to the number of particles used. An increase in the number of particles should thus improve its performance. Figure 5.1 showed the effect of increasing number of particles on the performance of the algorithms. Increasing the number of particles improved the estimation performance of the three algorithms in low-dimensions. In a low-dimensional setup, the three algorithms gave similar performance (p-value > 0.05), however it was noted that the RGAPF converged with fewer particles compared to the other two filtering algorithms.

The convergence of the RGAPF while using a small particle population size was explained using the schema theorem and the phenomenon of implicit-parallelism. Holland's schema analysis had showed that a GA while explicitly calculating the fitness of the N members of a population, implicitly estimates the average fitness of a much larger number of schemas by implicitly calculating the observed average fitness of schemas with instances in the population. It does this without needing any additional memory or computation time beyond that needed to process the N members of the population. Holland showed that for a population of N members, the GA implicitly process instances of order $O(N^3)$.

Hence the particle filter would require lesser number of particles and this was exactly the observations of our particle scaling experiment.

Our next test was increasing the dimensions of the state. Increasing the dimensions led to the collapse of the two benchmark algorithms, the PLA and the PF-LW. However the RGAPF was able to maintain its performance, and was successfully able to scale to higher-dimensions. The scalability of the RGAPF to higher-dimensions was an implication of the building-block hypothesis. The RGAPG was able to select particles with above average fitness and the recombination operator was able to combine them onto a single string.

In the first experiment we showed that our proposed RGAPF was successful in scaling to higher-dimensions and the results of this experiment strengthened our belief in the hypothesis that GA theory can be used to address and study particle filtering algorithms.

8.1.2 Experiment 2

The hypothesis used to address the sample impoverishment in particle filters was based on the building-block hypothesis that holds the recombination operator responsible for constructing higher-order schemata by utilizing lower-order building-blocks. To further strengthen our argument that recombination is responsible for the building-block like effects and it does not act like a variable rate mutation operator we carried out a second experiment in chapter 6.

To test this argument we devised a particle filter that was mutation-only; however the mutation-rate of this particular particle filter was not pre-defined. In our experimental setup first a recombination based RGAPF would run for an iteration, the before and after population variance would be calculated and this would be used in the mutation-only particle filter as a stopping criteria.

If the argument that recombination is a variable rate mutation operator is indeed correct then both the algorithms would achieve similar performance. However the results of our experiments supported the building-block hypothesis.

Figure 6.3 showed the performance comparison of the RGAPF with the mutation-only particle filter. The mutation-only particle filter gave incorrect estimates and its estimates were inconsistent. Furthermore the RGAPF performance was similar to the performance achieved in the first experiment and it scaled efficiently to higher-dimensions.

Thus the results of this experiment were not able to refute the concept of the building-block hypothesis. The hypothesis of this thesis and its emphasis on recombination for being able to address the curse of dimensionality in particle filters thus became more plausible after the second experiment.

8.1.3 Experiment 3

The first and the second experiment showed the validity of the approach used in this thesis. The RGAPF used an arithmetic recombination operator in the first two experiments. In the field of real-coded GAs, many different recombination operators have been proposed. The third experiment focused on testing the performance of RGAPF using these different recombination operators. The recombination operators used were mean-centric, parent-centric and n-point recombination, while the arithmetic recombination was used as a benchmark.

The results of this experiment showed that an RGAPF with a *uni-modal normal distribution* crossover (UNDX) consistently outperformed other recombination operators. The results of this experiment were opposite to the results obtained by Deb et al., in [DJA03]. In [DJA03], Deb et al., had showed that the parent-centric recombination, mPCX, was a better recombination operator compared to the UNDX. They showed that it converged quicker and its computational complexity was a lot lower compared to the UNDX. However Deb et al., had tested the mPCX on three optimization problems. The aim of the particle filter is to generate a particle population that represents samples from a distribution; hence all the particles within the population should be guided to an optimum representation of the posterior.

Based on the results of the third experiment we concluded that a UNDX operator essentially creates an offspring near the vicinity of the expected value of the density function. The whole GA layer within a particle filter would first sample multiple parents from a particle population; these parent particles would be selected based on their weights with the probability of particles with greater weights being sampled is greater compared to particles with lesser weights. These parent particles would then be used to calculate a mean-particle vector. Intuitively, the parent particles with greater weights are a good representation of samples from the posterior and hence their mean should be closer to the expected value of the posterior. A search within the vicinity of the mean could thus guide the particle population in the right direction, and hence the particle population will be updated iteration-by-iteration to remain diverse, yet closer to the actual posterior's expected value.

8.1.4 Experiment 4

The third experiment showed that the UNDX operator consistently outperformed other recombination operators in a particle filtering set-up. The conclusion drawn from this experiment was that since selection and mean-centric recombination create off-spring within the vicinity of the expected value of the posterior, it was able to guide the particle population towards the correct distribution.

It was then discussed that the UNDX operator is computationally expensive to implement compared to other recombination operators. In chapter 8 that UNDX operator was analysed and a similar mean-centric operator, *the mean-centric Gaussian recombination operator* (MCGR) was proposed. The MCGR is of linear complexity and provided estimates similar in accuracy to the UNDX operator.

8.1.5 Experiment 5

The first four experiments were carried out using simulated price time series, however for the fifth and final experiment, real end of day price time-series data form the London Stock Exchange was used. The pricing models were first calibrated using MCMC techniques and the result of the MCMC analysis was used as a benchmark.

The performance comparison of three RGAPFs, with UNDX, MCGR and arithmetic recombination was then carried out, and a hybrid CMA-ES particle filter taken from [SH12] was also added in this comparison. The results of this experiment showed a resemblance to the results obtained when the simulated data was used, another key observation was a similar performance provided by the hybrid CMA-ES particle filter with the RGAPF using UNDX and MCGR. The final experiment ended with the following key conclusions:

- GA theory can be used to address issues in particle filtering algorithms.
- The recombination operator in an RGAPF enables it to scale to higher dimensions.
- The UNDX operator outperforms other recombination operators in a particle filtering setup.
- The results of our experiments are also valid on real stock price data.
- A hybrid CMA-ES particle filter provides similar performance compared to a RGAPF, hence further research is required to analyse particle filters using ES theory.

8.2 Future Direction of Research

In this thesis the focus was on high-dimensional filtering applied to the stochastic volatility estimation of common stocks. Our approach was based on exploiting the similarities between particle filters and real-coded genetic algorithms and then using GA theory to address the issues faced by particle filters. The test problem under investigation had a uni-modal posterior distribution however according to real-coded genetic algorithm theory it may face the issue of ‘blocking’ if the state-space is multi-modal. Furthermore since the issue of scalability to higher dimensions was based on combining building-blocks using recombination, further research into creating population sizing models needs to be carried out to ensure an optimum supply of building-blocks to the RGAPF. These future avenues for research are mentioned in the next two subsections.

8.2.1 Evaluation of Performance under Blocking

Virtual characters and alphabets provide a useful perspective from which to view the convergence mechanisms of rGAs. According to the rGA theory, one-dimensional basin features are selected early in the GA dimension-by-dimension and the collection of virtual alphabets thus selected is used in subsequent recombinative-selective search. This mechanism seems to side step the precision and aliasing problems that may occur when low-cardinality codes are used by allowing rGAs to adaptively select their own alphabets.

The use of rGAs may have some limitations when the posterior density function of the state process is multi-modal. Goldberg in [Gol93] stated that rGAs can be thwarted from finding the global optimum by a phenomenon called ‘blocking’. In some multi-modal high-dimensional cases the virtual characters will be prevented from finding the global optimum because selection and mutation will only be able to perform hill climbing and will get stuck on one of the two local optima guarding the global optimum. The global-optima in this case is said to be blocked. Goldberg noted that there are limits that must be recognized and the rGA design should be modified on a case by case basis. Goldberg also noted that averaging recombination operators are unlikely to be of much practical help in overcoming blocking. Although there are many variations of averaging recombination operators, each of these theoretically offers some hope against blocking because each can jump somewhere very different from current parents but the chance of hitting a useful target is quite small.

Goldberg suggested a form of mutation that jumps anywhere within the allowable parameter interval to overcome blocking. In theory, since the GA is no longer restricted to the asymptotic hill climbing behaviour of selection and creeping mutation, it can get unstuck. Unfortunately, such operators are very disruptive and can only be used with low probability.

Additionally for a jump-mutated offspring to survive it had better jump to a point at or above the current average fitness. Point slices through the likeliest individuals can be checked to determine whether such jumps are going to do much good. The virtual characters are located where they are because the feature or features associated with that interval are of sufficient breadth and height to stick out above the crowd. Jumping to an above-average, unrepresented point that can hill climb to the global optimum is an unlikely event. In other words, the line search of jump mutation is likely to fail because good features those are not close to already-represented virtual characters.

The RGAPF may encounter these issues in a multi-modal state-space. Further research needs to be carried out to evaluate the performance of RGAPFs in estimating multi-modal density functions to test for the presence of the phenomenon of blocking and propose different operators to circumvent these issues if observed.

8.2.2 Research into a Population Sizing Model

The work done in this thesis is based on the ability of the recombination operator to combine building-blocks onto single strings. However further work needs to be carried out to ensure that an optimal number of building blocks are present inside the population.

In rGAs the population size that guarantees an optimal solution quickly has generally been perceived as one of the most important factors [GSL01]. All the studies have been performed under the assumption that the population is large enough to accommodate the actual dynamics of rGAs. That is, there is a fair measure of uncertainty when it comes to rGAs. Inevitably, rGA practitioners have to determine the population size without the necessary confidence. There are two approaches to the problem, one spatial and the other temporal [GSL01]. The spatial approach estimates the population size with a view to ensuring that a sufficient number of building-blocks with enough diversity are present to start with. The temporal approach assumes the existence of diversity-generating operators such as recombination and mutation that guarantee the required building-block diversity on a proper time scale. Many researchers have investigated the problem of supply of building-blocks for GAs under the assumption that the size of the string alphabet is finite. Holland [Hol75] estimated the number of building-blocks that receive at least a specified number of trials using Poisson distribution. Goldberg in [Gol89b] calculated the same quantity more accurately using binomial distribution and investigated its effects on population size in serial and parallel computation. Reevies in [Ree93] proposed a population sizing model for supply with alphabets of fixed cardinality.

Employing a spatial approach, Goldberg et al. in [GSL01] developed two facet wise models for ensuring building-block supply in the initial population. They also estimated the population size required to ensure the presence of all raw building-blocks with a tolerance in regard to fixed-length strings from alphabets of arbitrary cardinality. Sastry et al. in [SOGH03] also analysed building-block supply for genetic programming, along the lines of [GSL01].

To bring the population-sizing model to completion, decision making model (between competing building-blocks) must be considered. Holland [Hol75] studied the (k-armed) bandit problem as a theoretical motivation for GAs. Macready and Wolpert in [MW98] showed a mathematical flaw in Holland's analysis and provided an analytically simple bandit model that is directly applicable to optimization theory. De Jong in [Jon75] proposed a population-sizing equation based on the signal as well as noise characteristics of the k-armed bandit problem. Although the result explicitly exhibited the role of signal-to-noise ratio in estimating population size, the result was unverified and ignored [GDC92]. Goldberg and Rudnick [GR91] developed the first population-sizing equation based on the variance of fitness. Goldberg et al. in [GDC92] enhanced the equation as a conservative bound on the quality of GAs. The population-sizing equation permits accurate statistical decision making among competing building blocks. The population-sizing relation conservatively bounds the actual accuracy of GA convergence as long as all major sources of noise (i.e., collateral noise) are considered in the sizing calculation. Harik et al., in [HPGM99] also developed a population-sizing equation by incorporating building-blocks supply model with decision making model. It exploits similarity between the classical random walk problem – the gambler's ruin problem in particular and the selection mechanism of GAs for determining an adequate population size that guarantees a solution of the desired (target) quality. Ahn and Ramakrishna in [AR02] further enhanced and generalized the population-sizing equation in [HPGM99]. It can accurately estimate the population size required for achieving a desired quality of solution without any statistical information such as signal or collateral noise of competing building-blocks. Thus the importance of an optimal population is evident by the work that has already been done for optimization problems.

The success of the RGAPFs in high-dimensions was shown to be due to the combination of building-blocks that ensured that the particle filter was able to scale to higher-dimensions. Hence, research into optimum population sizing models is of utmost importance to ensure their success in high-dimensions. Along the lines of the work recounted above, further research is required to propose appropriate models for particle filters.

8.3 Summary

This chapter concluded the research carried out and laid the foundations of the future direction for further research. All the important concepts mentioned in this thesis and the results of each experiment were then summarized. This chapter was divided into two main sections. The first section summarized all the work done in this these. It started with a brief introduction to sequential Monte Carlo methods and the issues encountered when they are implemented in high-dimensional spatial systems. The approach of this research was then outlined. The premise of our approach is that if particle filters and genetic algorithms are similar then genetic algorithm theory can be used to assess the performance of particle filters. Based on this similarity experiments were conducted. The conducted experiments and their main observations were then mentioned.

The second section of this chapter mentioned further avenues of research. The phenomenon of 'blocking' and 'population sizing' from real-coded genetic algorithm were listed as two important topics that need to be looked into as they may have important implications on the performance of the RGAPFs.

The main text of this thesis concludes at this point. The appendix on the next page consists of the complete results of the experiments that were carried out in this thesis.

Bibliography

[AK77] H. Akashi and H. Kumamoto. Random Sampling Approach to State Estimation in Switching Environment. *Automation*. Pages 413- 429, 1977.

[AMGC02] S. Arulampalam, S. Maskell, N. Gordon and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, vol. 50. Pages 174-188, 2002.

[And99] S. L. Anderson. A Monte Carlo implementation of the nonlinear filtering problem to produce ensemble assimilations and forecasts. *Monthly Weather Review*, vol 127. Pages 2741–2758, 1999.

[AR00] S. Arulampalam, B. Ristic. Comparison of the Particle Filter with Range Parameterised and Modified Polar EKF for Angle-Only Tracking, *Signal and Data Processing of Small Targets*, vol. 4048. Pages 288–299, 2000.

[AR02] C.W. Ahn and R.S. Ramakrishna, A genetic algorithm for shortest path routing problem and the sizing of populations, *IEEE Transactions of Evolutionary Computation.*, vol.6, no.6. Pages 566–579, 2002.

[BBL08] T. Bengtsson, P. Bickel and B. Li, Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems, *IMS Collections. Probability and Statistics: Essays in Honour of David A. Freedman*, *Institute of Mathematical Statistics*, Vol. 2. Pages 316–334, 2008

[BD01] H.G. Beyer and K. Deb. On self-adaptive features in real-parameter evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 5(3). Pages 250–270, 2001.

[Bel61] R. Bellman. Adaptive Control Processes: A Guided Tour. *Princeton University Press*. 1961.

[BSN03] T. Bengtsson, C. Snyder, and D. Nychka. Toward a nonlinear ensemble filter for high-dimensional systems. *J. Geophys. Res*., 108 (D24). Pages 8775–8785, 2003

[CC04] Y. Cheng and J. L. Crassidis, Particle filtering for sequential spacecraft attitude estimation. *Collection of Technical Papers - AIAA Guidance, Navigation, and Control Conference*, 2004.

[CD02] D. Crisan and A. Doucet. A survey of convergence results on particle filtering methods for practitioners. *IEEE Transactions on Signal Processing*,50(3). Pages 736–746, 2002.

[CM03] J. Crassidis and F. L. Markley. Unscented filtering for spacecraft attitude estimation. *Journal of Guidance, Control, and Dynamics*. 2003.

[DAJ02] K. Deb, A. Anand, and D. Joshi. A computationally efficient evolutionary algorithm for real-parameter optimization. *Evolutionary Computation*, vol.10, no.4. Pages 371–395, 2002.

[DFG01] A. Doucet, J. Freitas, and N. Gordon. Sequential Monte Carlo methods in Practice. *Springer-Verlag, New York*. 2001.

[DJA02] K. Deb, D. Joshi and A. Anand. Real-Coded Evolutionary Algorithms with Parent-Centric Recombination. *Evolutionary Computatio. CEC '02. Proceedings of the 2002 Congress on (Volume:1)*. Pages. 61 – 66, 2002.

[ES07] E. A. Eiben and J.E. Smith. Introduction to Evolutionary Computing. *Springer (Natural Computing Series)*. Sep 2003.

[ES93] L.J. Eshelman and J.D. Schaffer J.D. Real-coded genetic algorithms and interval schemata. *Foundation of Genetic Algorithm* II. Pages 187-202, 1993.

[Eve07] G. Evensen. Data assimilation: The ensemble Kalman filter. *Springer-Verlag, Berlin, Heidelberg*. 2007.

[FA00] D. B. Fogel and R.W. Anderson, Revisiting Bremermann's Genetic Algorithm: I. Simultaneous Mutation of All Parameters. *Applications and Science of Computational Intelligence IV* . 2000.

[Far66] I. Farrell. Attitude determination by Kalman filtering. *NASA, Contractor Report CR*-598, 1966.

[GDC92] D.E. Goldberg, K. Deb, and J.H. Clark. Genetic algorithms, noise, and the sizing of populations. *Complex Systems,* vol.6, no.4, pages.333–362, 1992.

[Gol89] D.E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. *Addison Wesley, Reading, MA*. 1989.

[Gol89b] D.E. Goldberg, Sizing populations for serial and parallel genetic algorithms. *Proc.Third International Conference on Genetic Algorithms*. Pages.70–79, 1989.

[Gol91] D.E. Goldberg. Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex Systems,* vol.5. Pages 139–167, 1991.

[GR91] D.E. Goldberg and M. Rudnick. Genetic algorithms and the variance of fitness. *Complex Systems,* vol.5, no.3. Pages 265–278, 1991.

[GSL01] D.E. Goldberg, K. Sastry, and T. Latoza. On the supply of building blocks. *GECCO'01*. Pages.336–342, 2001.

[GSS93] N. Gordon, D. Salmond and A. Smith. A novel approach to nonlinear/non-Gaussian Bayesianstate estimation. *In lEE Proceedings on Radar and Signal Processing,* vol. 140. Pages 107-113, 1993.

[Gus10] F. Gustafsson. Particle Filter Theory and Practice with Positioning Applications. *IEEE Aerospace and electronic systems magazine*, (25), 7. Pages 53-81, 2010.

[Han70] J. Handshin. Monte Carlo techniques for prediction and filtering of nonlinear stochastic processes. *Automatica,* 6, 555. 1970.

[Hau12] A.J. Haug. Bayesian Estimation and Tracking: A Practical Guide. *Wiley Publishing.* May 2012

[HH98] R.L. Haupt and S.E. Haupt, Practical Genetic Algorithms, John Wiley & Sons, 1998.

[HLM98] F. Herrera, M. Lozano and J.L. Verdegay. Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis. *Artificial Intelligence Review* 12(4). Pages 265–319, 1998.

[HMP54] J. Hammersley and K. Morton. Poor man' s Monte Carlo. *Journal of the Royal Statistical Society*, Series B, 16, 23. 1954

[Hol75] J. Holland. Adaptation in Natural and Artificial Systems. *University of Michigan Press, Ann Arbor*. 1975.

[HPGM99] G. Harik, E. Cant´u-Paz, D.E. Goldberg, and B.L. Miller. The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Evolutionary Computation*, vol.7, no.3. Pages 231–253, 1999.

[Hu01] J. C. Hull. Options, Futures and Other Derivative. *Prentice Hall (Mathematics, Finance and Risk*). Oct 2005.

[Hus12] Muhammad Shakir Hussain. Real Coded Genetic Algorithm Particle Filter for Improved Performance. *IPCSIT* vol. 25 (2012)

[IB96] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. *Europe Conference On Computer Vision (ECCV)*. Pages 343–356, 1996.

[IB98] M. Isard and A. Blake. Condensation-Conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29, 1. Pages 5-28, 1998

[Jay03] E.T. Jaynes. Probability Theory: The Logic of Science. *Cambridge University Press*. June 2003

[Jo08] M. Joshi. The Concepts and Practice of Mathematical Finance. *Cambridge University Press (Mathematics, Finance and Risk)*. 30 Oct 2008.

[Jon75] K.A. De Jong, An Analysis of the Behavior of a Class of Genetic Adaptive Systems. *Doctoral Dissertation, University of Michigan, Ann Arbor, MI*, 1975.

[JPR95] E. Jacquier, N.G. Polson and P. Rossi. Stochastic Volatility - Univariate and Multivaiate extensions. May 1995.

[Ju98] S.Julier, A Skewed Approach to Filtering. *Signal and Data Processing of Small Targets. SPIE.* Vol 3373. Pages 271-282, 1998.

[Kal60] R. Kalman. A new approach to linear filtering and prediction problems. *Transactions of Journal Basic Engineering, ASME Series D,* 82. Pages 35-45, 1960.

[KD03] J. H. Kotecha and P. M. Djuric. Gaussian particle filtering. *IEEE Transactions of Signal Processing vol. 51*. Pages 2593–2602, Oct. 2003.

[KFZ05] Kwok, N.M, Gu Fang ; Weizhen Zhou. Evolutionary particle filter: re-sampling from the genetic algorithm perspective. *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference*. Pages: 2935 - 2940

[Kit96] G. Kitagawa. Monte Carlo filter and smoother for non-Gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*. Pages 1-25, 1996

[Ko92] J.R. Koza. Genetic Programming. *The MIT press, Cambridge*.1992

[KOK98] H. Kita, I. Ono, and S. Kobayashi, Theoretical analysis of the unimodal normal distribution crossover for real-coded genetic algorithms. Proc. *IEEE International Conference on Evolutionary Computation.* Pages 529–534, 1998.

[Kra03] E. Kraft. A quaternion-based unscented Kalman filter for orientation tracking. *Proceedings of the Sixth International Conference on Information Fusion,* 2003.

[KSH00] T. Kailath, A. Sayed, and B. Hassibi. Linear Estimation. Information and System Sciences Series. Upper Saddle River, N J : Prentice-Hall , 2000.

[Lee03] P.J. van Leeuwen. A variance-minimizing filter for largescale applications. *Monthly Weather Review 131*. Pages 2071–2084, 2003.

[Lee09] P.J. van Leeuwen. Particle filtering in geophysical systems. *Monthly Weather Review,* 137. Pages 4089–4114, 2009.

[LW90] S.L. Lu and K. Wohn. Estimation of General Rigid Body Motion from a long Sequence of Images. *Technical Report, Department of Computer Science, University of Pennsylvania*. 1990.

[LW01] J. Liu.and M. West. Combined parameter and state estimation in simulation-based filtering. *Sequential Monte Carlo in Practice*, *Springer.* Pages 197-217, July 2001, 2001.

[Mar04] F. L. Markley. Multiplicative vs. additive filtering for spacecraft attitude determination. *Dynamics and Control of Systems and Structures in Space,* 2004.

[MFH92] M. Mitchell, S. Forrest and J.H. Holland. 1992. The royal road for genetic algorithms: Fitness landscapes and GA performance. *Toward a Practice of Autonomous Systems:Proceedings of the First European Conference on Artificial Life.* MIT Press. 1992

[MHF94] M. Mitchell, S. Forrest and J.H. Holland. When will a genetic algorithm outperform hill climbing? *Advances in Neural Information Processing Systems. Morgan Kaufmann*. 1994.

[Mit98] M. Mitchell. An Introduction to Genetic Algorithms. *Complex Adaptive Systems* [Paperback], April 1998.

[Mo98] P. Del Moral. Measure-valued processes and interacting particle systems. Application to nonlinear filtering problems. *Annals of Applied Probability,* 8(2). Pages 438–495, 1998.

[MR01] K. Murphy and S. Russell. Rao-Blackwellised particle filtering for dynamic Bayesian networks. *Sequential Monte Carlo methods in practice*, Springer-Verlag, New York. Pages. 500-512, 2001.

[MT05] M. Montemerlo and S. Thrun. The FastSLAM Algortihm for Simultaneous Localization and Mapping. *Springer Tracts in Advanced Robotics*. 2005.

[MW04] R. van der Merwe and E. Wan. Sigma-point Kalman filters for integrated navigation. *60th Annual Meeting of the Institute of Navigation*, 2004.

[MW98] W.G. Macready and D.H. Wolpert. Bandit problems and the exploration/ exploitation tradeoff. IEEE Transactions of Evolutionary Computation, vol.2, no.1. Pages 2–22, 1998.

[MYBMZ01] J. Marins, X. Yun, E. Bachmann, R. McGhee, and M. Zyda. An extended Kalman filter for quaternion-based orientation estimation using MARG sensors. *IEEE International Conference on Intelligent Robots and Systems*. 2001.

[OHSZ04] E. Ott, B.R. Hunt, I. Szunyogh, A.V. Zimin, E.J. Kostelich, M. Corazza, E. Kalnay, D.J. Patil, and J.A. Yorke. A local ensemble transform Kalman filter for atmospheric data assimilation. Pages 415–428, 2004.

[OK97] I. Ono and S. Kobayashi. A real-coded genetic algorithm for function optimization using unimodal normal distribution crossover. *7th International Conference on Genetic Algorithms.* Pages 246–253, 1997.

[Pan05] Juan Jose Pantrigo. Combining Particle Filter and Population-based Metaheuristics for Visual Articulated Motion Tracking. *Electronic Letters on Computer Vision and Image Analysis.* 2005. Vol5, no.3.

[PHRK07] S. Park, J. Hwang, K. Rou, and E. Kim. A New Particle Filter Inspired by Biological Evolution: Genetic Filter. *World Academy of Science, Engineering and Technology. International Journal of Electrical, Electronic Science and Engineering* Vol:1 No:9, 2007

[QMG08] P.B. Quang, C. Musso, F. Gland. An Insight into the Issue of Dimensionality in Particle Filtering. 2008.

[QP94a] X. Qi and F. Palmieri. Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space, Part I: Basic properties of selection and mutation. *IEEE Transactions on Neural Networks*, vol.5, no.1. Pages 102–119, January 1994.

[QP94b] X. Qi and F. Palmieri. Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space, Part II: Analysis of the diversification role of crossover. *IEEE Transactions on Neural Networks*, vol.5, no.1. Pages 120–129, January 1994.

[RAG04] B.Ristic, S.Arulampalam, N.Gordon. Beyond the Kalman Filter: Particle Filters for Tracking Applications. *Artech House Radar Library*. January 2004.

[RB06] D. Raggi and S. Bordignon. Sequential Monte Carlo Methods for Stochastic Volatility Models with Jumps. 2006.

[Ree93] C. Reeves. Using genetic algorithms with small populations. *Fifth International Conference on Genetic Algorithms*. Pages 92–99, 1993.

[RN99] Miguel Rocha , José Neves . Preventing Premature Convergence to Local Optima in Genetic Algorithms via Random Offspring Generation. *12th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems IEA/AIE-99, Cairo, Egypt,* May 31 - June 3, 1999.

[RR56] M. Rosenbluth and A. Rosenbluth. Monte Carlo calculation of the average extension of molecular chains. *Journal of Chemical Physics*. Pages 590-613, 1956.

[RSB99] S. Roumeliotis, G. Sukhatme, and G. Bekey. Circumventing dynamic modeling: Evaluation of the error-state Kalman filter applied to mobile robot localization. *IEEE International Conference on Robotics and Automation*. 1999.

[Rub81] B.Y. Rubinstein. Simulation and the Monte Carlo Method. *New York: Wiley & Sons*. 1981.

[SBBA08] C. Snyder, T. Bengtsson, P. Bickel, J. Anderson. Obstacles to High-Dimensional Particle Filtering. *American Meteorological Society*. May 2008.

[SDFG01] A. Smith, Arnaud Doucet, Nando de Freitas and Neil Gordon. Sequential Monte Carlo Methods in Practice. *Information Science and Statistics, Springer*. July 2001.

[SH12a] R.E. Smith and M.S. Hussain. Hybrid Metaheuristic Particle Filters for Stochastic Volatility Estimation. *GECCO 2012*. Pages 1167-1174, 2012.

[SH12b] Robert Smith and Muhammad Shakir Hussain. Genetic Algorithm Sequential Monte Carlo Methods For Stochastic Volatility And Parameter Estimation. Proceedings of the World Congress on Engineering 2012 Vol I WCE 2012, July 4 - 6, 2012.

[Siv06] D. Silva. Data Analysis : A Bayesian Tutorial. *Oxford Science Publications*. July 27, 2006.

[SOGH03] K. Sastry, U.M. O'Reilly, D.E. Goldberg, and D. Hill. Building block supply in genetic programming. *IlliGAL Report no.200312, University of Illinois at Urbana-Champaign*. April 2003.

[Spe98] W. Spears. The Role of Mutation and Recombination in Evolutionary Algorithms. PhD thesis, George Mason University. 1998.

[Str60] R. L. Stratonovich, Conditional Markov Process Theory. *Theory Prob. Appl. (USSR).* Vol. 5. Pages 156-178, 1960.

[Tal09] G.Al-Talbi, Metaheuristics: From Design to Implementation. *Wiley Series on Parallel and Distributed Computing*. 10 July 2009.

[TFBD00] S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust monte carlo localization for mobile robots. *Technical Report CMU-CS-00-125,* Carnegie Mellon University, Computer Science Department, Pittsburgh, PA, 2000.

[TGS01] S. Tsutsui, D.E. Goldberg and K. Sastry. Linkage learning in real-coded GAs with simplex crossover. *Proceedings of Evolutionary Algorithms*. Pages 73–84, 2001.

[Th00] S. Thrun. Probabilistic algorithms in robotics. *Technical Report CMU-CS-00-126*, Carnegie Mellon University, Computer Science Department, Pittsburgh, PA. 2000.

[VMC95] H.M. Voigt, H. Muhlenbein and D. Cvetkovic. Fuzzy recombination for the Breeder Genetic Algorithm. *Proceedings of the Sixth International Conference on Genetic Algorithms*. Pages 104–111, 1995.

[Whi93] L.D. Whitley. Foundations of Genetic Algorithms .*Morgan Kaufmann*, edition 2. 1993.

[WV95] L.D. Whitle and M.D. Vose. Foundations of Genetic Algorithms. *Morgan Kaufmann*, 1995.

[ZM04] Q. Zhang and H. Muhlenbein. On the convergence of a class of estimation of distribution algorithms. *IEEE Trans. Evolutionary Computation*, vol.8, no.2. Pages 127–136, 2004.