Portfolio Optimisation with Sequential Monte Carlo

Yow Tzu Lim

MSc in Statistics

Imperial College, London

Department of Mathematics

August 10, 2014

Abstract

As computer system size and complexity grow, formulating effective policies require more sophistication. There are many risk factors need to be considered, some of which may be in conflict. Inevitably, unpredictable circumstances that demand decisions will arise during operation. In some cases an automated response may be imperative; in other cases these may be ill-advised. Manual decisions made to override the default ones, in a sense, redefining the underlying policy. This matter is further complicated in highly dynamic operational environments like mobile ad-hoc networks, in which the risk factors may be changing continually. Thus, security policies must be able to change and adapt to the operational needs.

This study investigates the potential of evolutionary algorithms as a tool in determining the optimal security policies that suit such environments. This thesis reviews some fundamental concepts in related domains. It presents three applications of evolutionary algorithms in solving problems that are of direct relevance. These include the inference of security policies from decision examples, the dynamic adaptation of security policies and the optimisation of security policies for a specific set of missions. The results show that the inference approaches based on evolutionary algorithms are very promising. These approaches are also sufficiently generic to be used as general dynamic classification algorithms.

The thesis concludes with an evaluation on the work done, the extent to which the work justify the thesis hypothesis and some possible directions on how evolutionary algorithms can be applied to address a wider range of relevant problems on the domain of concern.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my gratitude to all who have provided support in accomplishing this thesis. I would like to thank my supervisor, Dr. Nikolas Kantas, for the opportunity given to carry out this research and also for his guidance, expertise and encouragement during the course of this research. I would like to thank the support staff, especially XYZ and ABC, who have maintained the Tesla grid servers, on which my experiments have been carried out.

I would also like to express my appreciation to my wife Ka Ki Lai for her endless support and care in my personal life during the course of this master degree study. I would like to thank God Almighty, who has given me the strength and peace to work hard on this project. I would like to express my gratitude to all who have provided support in accomplishing this thesis.

Declaration

The work contained in this thesis is my own work unless otherwise stated.

I am the primary author of all work reported in this thesis. Advice on specific aspects of the work was provided by my supervisor, Dr. Nikolas Kantas.

Chapter 1

Introduction

Resource allocation is a common challenge for every investor. In the investment decision making process, investors decide how much resources are allocated to different investable assets to form a portfolio with the aim to optimise the performance of the overall portfolio is better off than any other according to some criterion. The criterion can be different to the investors. (Some investors may have different considerations such as tax considerations and legal restriction on investment assets or holding periods.)

Two common objectives, often contradicting, are the financial return and the investment risk. The Markowitz's modern portfolio theory [1] proposes a portfolio selection framework. In Markowitz's model, it is assumed that investor attempt to maximize a portfolio's return and minimize the risk (measured by the variance of the portfolio return. Based on this criteria, the set of non-dominance portfolio is known as the *efficient porfolios*. Using variance as a risk measure has its limitation. Variance is a symmetric measure; an out performing asset (than the expected return) is deemed to as risky as an under perfoming one. Many alternative risk measurements have been proposed, e.g. Sortino ratio, Conditional Value at Risk (CVaR), etc. Refer [2] for details.

Surprisingly, there are some investment managers who have no interest on maximising their portfolio's return. Instead, the main objective of portfolio management for such a fund is simply track and replicate the exposure of a benchmark index as close as possible. These funds are attractive as it provides investors the exposure to the market, at the same time, minimal active management required makes the fund less vulnerable to change of management and has the advatanges of lower fees and taxes. The performance of these funds are often assessed in term of how well the fund tracks the benchmark index using some pre-defined metrics.

1.1 Technical Approach

Traditionally, portfolio optimization have been explored in an analytical fashion, adopting necessary assumption as necessary. This seems rather restrictive; there are many instances where numerical method has been used to derive an approximate or even more effective solution to the problem in question. For example, Monte Carlo technique is used to do integral, evolutionary techniques applied in engineering domains, etc.

Our approach to the problem in this thesis is a radical one. We view a portfolio optimisation as a *stochastic* control problem. We adopt the Bayesian view and treat tehse parameters as random variables. The objective is to find the sequence of control parameters that optimise the control objective defined in terms of portfolio return and financial risk. We investigate the potential of using SMCs as the means of determining the optimal strategy, or at least excellent, strategies for the portfolio optimisation problem in question. The main reason of choosing SMCs is its ability to carry out *sequential* update on the posterior distribution over time fit well with parameter inference in stochastic process. Moreoever, these techniques have achieved significant success in their applications on many domains. Of course, other heuristic search tecniques are also potentially applicable.

To investigate this approach, we first applied the technique on to a simpled determinstic reference model. This model is doubly useful. It demonstrates the concept nicely and serves as a basic model to allow us to gain further understanding on the tunable parameters. We then considered a simplied a market model with two assets: one risky asset with its price modelled as Brownian motion with drift, and one zero interest risk-free asset (constant) which has an analytical solution. Using SMCs, we search for the optimal strategy (the set of control parameters at each time point) that optimise against the optimisation criteria (a.k.a. reward) in terms of expected return over the investment period, and minimize the financial risk (variance of the return) using SMC techniques. The results are then evaluated against with the analytic solution described in [3].

1.2 Thesis hypothesis

Formally, the hypothesis of the thesis is stated as follows:

> Sequential Monte Carlo (SMCs) have the potential to be an effective means of searching the optimal strategy for portfolio problem.

We attempt to examine this hypothesis from three different perspectives:

1. Exploring the potential of SMCs in searching the optimal strategy from mean-variance criteria with constraints.

2. Exploring the sensitivity of the techniques in terms of the parameter settings.

3. Exploring the trade-off between the estimation accuracy, the complexity of the problem complexity and the computational efforts numerically and providing suggestions for real-world problem.

Given the time frame, we fully understand it is impossible to evaluate our approach on full scale strategy. The aim is to establish the plausibility or, at the very least, a greater understanding of the strengths and weaknesses of the above approach.

1.3 Thesis organisation

The subsequent chapters of this thesis are organised as follows:

- Chapter 2 reviews some fundamental concept in Monte Carlo method that are related to this thesis. It begins with a brief introduction to basic methods such as perfect Monte Carlo sampling, rejection sampling, importance sampling. It then details two common Markov Chain Monte Carlo (MCMC) techniques, namely Metropolis-Hastings and Gibbs Sampling. Lastly, it introduces the Sequential Monte Carlo (SMC) technique used in this thesis.

- Chapter 3 briefly review the state of the art of portfolio optisation problem. It then discusses how a portfolio problem can be transformed naturally into standard parameter estimation in Sequential Monte Carlo frameowrk.

- Chapter 4 details the toy experiment in which we attempt to use SMC to track a reference signal generated by a known synthetic model.

- Chapter 5 details the experiment in using SMC to infer the optimal control for portfolio that tracks real-world indices. In particular, we focus on the major stock indices accross the continient. It first discusses the market model that is used. It then details the problem form ulation. Lastly, it discusses the experimental results, in comparison to the theorical results.

- Chapter 6 concludes the thesis by evaluating the degree to which the hypothesis has been justfied and outlines potential work for the future.

Chapter 2

Monte Carlo Methods

Monte Carlo Methods have achieved significant success in its aplication to various domains in the last few decades. This chapter reviews some fundamental concept in Monte Carlo method that are related to this thesis. It first begins with a summary on the main concept of Bayesian inference. It then discusses some basic Monte Carlo methods such as perfect Monte Carlo sampling, rejection sampling, importance sampling. Lastly, it details the Sequential Monte Carlo (SMC) techniquesthat is used to do portfolio optimisation, along with various enhancement made to the framework.

2.1 Bayesian Inference

In Bayesian inference framework, each unknown parameter in the model is assumed to be random variable and is associated with a prior distribution that characterises the initial belief. The inference process is merely about updating the belief with new observable evidence in a systematic fashion using Bayes theorem.

Formally, let M be the Bayes model of interest, θ be the set of parameters of the model, $p(\theta \mid M)$ be the prior distribution (initial belief) and $p(x \mid \theta, M)$ be the likelihood (probability of observing an observation x given the model) then posterior distribution (updated belief) is given as follows:

$$p(\theta \mid x, M) = \frac{p(x \mid \theta, M)\ p(\theta \mid M)}{p(x \mid M)}$$
$$\propto p(x \mid \theta, M)\ p(\theta \mid M) \tag{2.1}$$
$$\text{posterior} \propto \text{likelihood} \times \text{prior} \tag{2.2}$$

This problem formulation is elegant, but there remains some subtle issues in practice. One particular issue is about the calculation of the normalisation constant $p(x \mid$

M) in (2.1), which demands us to be able to carry out the following integral analytically:

$$p(x \mid M) = \int p(x \mid \theta, M)p(\theta \mid M)\ d\theta \tag{2.3}$$

This is often infeasible. A often way to circumvent this requirement is by making use of conjugate prior that yields posterior distributions from the same family in an analytical fashion.

Instead of a closed form solution, the Method Carlo methods offer a numerical solution in estimating the integral using sampling technique. The need of calculating integral that does not posess analytic solution also arises in the marginalisation process of nuisance parameters, calculating expectation of a function, etc.

2.2 Perfect Monte Carlo

Consider the calculation the expectation of a function, I of the following form:

$$I = \mathbb{E}_p[f(x)] = \int f(x)p(x)\ dx \tag{2.4}$$

Assuming we are able to sample N independent and identically distributed (i.i.d.) samples of x from $p(\cdot)$, denote these as $\{x^{(i)}\}$ where $i \in \{1 \dots N\}$, a Monte Carlo estimate of I using the the point masses of the samples is:

$$\hat{I} = \frac{1}{N}\sum_{i=1}^{N} f(x^{(i)}) \tag{2.5}$$

This approximation can be informally described as discretization of the continuous distribution with *random* support. This estimate has been shown to be unbiased and converge almost surely to be unbiased and converge almost surely to I as $N \to \infty$ by the Law of Large number [4].

Morever, if the variance of $f(\cdot)$ is bounded ($\sigma_f^2 < \infty$), then the following central limit theorem holds:

$$\sqrt{N}(\hat{I} - I) \Longrightarrow N(0, \sigma_f^2) \tag{2.6}$$

as $N \to \infty$, where $\Longrightarrow$ denotes convergence in distribution [5]. The key point to note here is this converengence rate of $\frac{1}{\sqrt{N}}$ is independent of the dimensions of x. This is in constrast with any determinstic method that has a rate that decreases as the integral dimension increases [4]. This is the main advantage of Monte Carlo integration.

2.3 Rejection sampling

However, it is not always possible to sample directly from the distribution $p(\cdot)$. Suppose we can find an instrumental distribution (a.k.a. proposal distribution), $q(\cdot)$, that is easy to sample from and has the property such that $cq(x)$ dominates $p(x)$ for all x, i.e., $cq(x) \geq p(x) \geq 0$ for all x, then to get a random sample from $p(\cdot)$, we can first sample from $q(\cdot)$ instead and accept the sample with acceptance probability $\alpha(x) = \frac{p(x)}{cq(x)}$. If the sample is rejected, repeat the process until success. This algorithm is summarised in Algorithm 1[1].

Algorithm 1 Rejection Sampling

function RejectionSampling(N)
  $\mathcal{X} = \{\ \}$.
  repeat
    sample $x \sim q(\cdot)$.
    sample $u \sim \mathcal{U}(0, 1)$.
    if $u \leq \frac{p(x)}{cq(x)}$ **then**.
      $\mathcal{X} \leftarrow \mathcal{X} \cup \{x\}$.
    end if
  until len($\mathcal{X}$)=N.
  return $\mathcal{X}$.
end function

Looking at the acceptance ratio formula, it is not difficult to see that the optimal instrumental distribution, q^*, is the one that minimizes the space bounded by $cq(x)$ subject to the constraint that it still dominates the target density $p(x)$. As the dimension of x increases, this algorithm becomes very ineffecient because the acceptance ratio whic is essentially defined as the ratio of two embedded spaces tends towards zero. Therefore, many generated examples would be rejected.

2.4 Importance sampling

Instead of making a binary accept-reject decision on each sample, the key concept in important sampling is assign weighting to each sample (obtained from the instrumental distribution, $q(\cdot)$) based on how well the sample resembles the target distribution, $p(\cdot)$. More formally, assume we have an instrumental distribution, $q(\cdot)$ that has support that

[1]The set notation is used here, despite it is actually a collection that allows duplicates. Therefore, the $\cup$ operation is really joining two collections together

includes $p(\cdot)$, we can re-write (2.4) as:

$$\begin{aligned} I &= \int f(x)\frac{p(x)}{q(x)}q(x)\ dx \\ &= \int f(x)w(x)q(x)\ dx \\ &= \mathbb{E}_q[f(x)w(x)] \end{aligned} \tag{2.7}$$

where $w(x)$ is commonly referred as the importance weight. This reformulation leads to the following Monte Carlo estimate of I:

$$\begin{aligned} \hat{I} &= \frac{\frac{1}{N}\sum_{i=1}^{N}\tilde{w}(x^{(i)})f(x^{(i)})}{\frac{1}{N}\sum_{j=1}^{N}\tilde{w}(x^{(j)})} \\ &= \sum_{i=1}^{N}\frac{\tilde{w}(x^{(i)})}{\sum_{j=1}^{N}\tilde{w}(x^{(j)})}f(x^{(i)}) \\ &= \sum_{i=1}^{N}\hat{w}(x^{(i)})f(x^{(i)}) \end{aligned} \tag{2.8}$$

where $\tilde{w}(x^{(i)}) = \dfrac{p(x^{(i)})}{q(x^{(i)})}$ and $\hat{w}(x^{(i)}) = \dfrac{\tilde{w}(x^{(i)})}{\sum_{j=1}^{N}\tilde{w}(x^{(j)})}$ are referred to as unnormalised and normalised importance weight respectively [6]. This estimate is biased as it consists of the ratio of two estimates, yet it is still asymtoptically consistent.

To obtain samples from the target distribution, $p(\cdot)$, an additional resampling step can be introduced. In the first step, we draw a set of samples $\{\tilde{x}^{(i)}\}$ from the instrumental distribution and compute their associated normalised importance weights, $\hat{w}(\tilde{x}^{(i)})$. In the resampling step, we draw the final sample set, $\{x^{(i)}\}$ from this intermediate set of samples by taking into the importance weights. This algorithm is summarised in Algorithm 2

There are many ways of implementing the resampling stage. A simple direct implementation is to select the sample from the intermediate stage according to a Multinomial distribution with the success probability parameter set to the vector of normalised weights, $\hat{w}(x^{(i)})$, i.e., the chance of a sample point being replicated is proportional to its weight. This resampling step however introduces extra variance to the estimators, yet this can be a crucial step in the sequential scheme that we shall look in the following section to avoid sampling degeneracy over time.

2.5 Sequential Monte Carlo

To motivate why Sequential Monte Carlo is useful, consider a target distribution of interest $p(x_{1:n})$, and for simplicity, assuming we can sample directly from the distribution $p(x_{1:n})$, the minimal computational complexity of the sampling scheme would

Algorithm 2 Importance Sampling

function ImportanceSampling(N)
  $\tilde{\mathcal{X}} = \{\ \}$
  repeat
    sample $\tilde{x} \sim q(\cdot)$.
    $\tilde{\mathcal{X}} \leftarrow \tilde{X} \cup \{\tilde{x}\}$.
  until len($\tilde{X}$)=N.
  calculate importance weights, $\hat{w}(\tilde{x}^{(i)}) = \dfrac{\tilde{w}(\tilde{x}^{(i)})}{\sum_{j=1}^{N} \tilde{w}(\tilde{x}^{(j)})}$.
  $\mathcal{X} = \{\ \}$.
  repeat
    sample x from $\tilde{\mathcal{X}}$ according to the importance weights, $\hat{w}$.
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{x\}$.
  until len($\mathcal{X}$)=N.
  return $\mathcal{X}$.
end function

be at least linear in n. Sequential Monte Carlo (SMC) provides a way to obtain samples for each sequential time step in a *fixed* amount of computational time in Hidden Markov Models (HMMs). We shall begin with a brieft introduction on HMMs that is cruicial to understand SMC in the next section. Refer [6] for further details of inference techniques for HMMs in general.

2.5.1 Hidden Markov Models

HMMs can be seen as a class of models that consist of two related processes: an underlying Markov process, X_t, which is the target process of interest, and a observable process, Y_t, which its state can be measured and therefore provides some information about X_t. Moreover, it is assumed that these two processes have conditional independence properties as shown using the graphical model representation in Figure 2.1. These properties can be summarised as follows:

$$\begin{aligned} p(x_t \mid x_{0:t-1}) &= f(x_t \mid x_{t-1}) \\ p(y_t \mid x_{0:t}, y_{0:t-1}) &= g(y_t \mid x_t) \end{aligned} \tag{2.9}$$

where $f(x_t \mid x_{t-1})$ is the transition density and $g(y_t \mid x_t)$ is the likelihood.

This class of models are designed to model evolving systems that output some observable events over time, which depends on the internal state of the system in a online manner. This imposes an implicit requirement on computation perspective: the

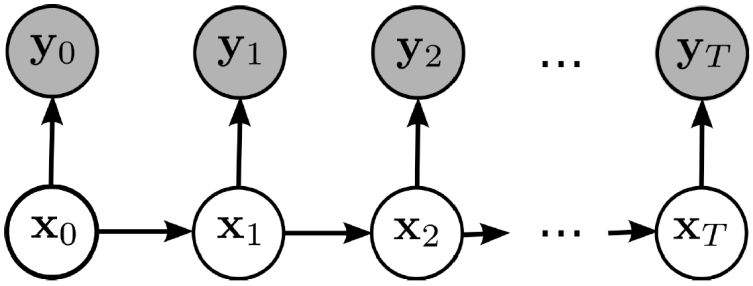

Figure 2.1: Hidden Markov Models

estimate calculation cost should remain constant over time, i.e., the calculation cost does not increase with the increasing number of states.

Arguably, the most common inference problem in HMMs is the smoothing distribution, $p(x_{0:t} \mid y_{0:t})$, that is estimating the states $x_{0:t}$ based on the sequence of observations up to time t, $y_{0:t}$. Using Bayes rules, we can write the density of the distribution of interest in the recursion form as follows:

$$\begin{aligned} p(x_{0:t} \mid y_{0:t}) &= \frac{p(x_{0:t}, y_{0:t})}{p(y_{0:t})} \\ &= \frac{p(x_{0:t-1}, y_{0:t-1}) f(x_t \mid x_{t-1}) g(y_t \mid x_t)}{p(y_{0:t})} \\ &= p(x_{0:t-1} \mid y_{0:t-1}) \frac{f(x_t \mid x_{t-1}) g(y_t \mid x_t)}{p(y_t \mid y_{0:t-1})} \end{aligned} \tag{2.10}$$

This recursion is often re-written into two separate steps: the prediction step (the estimation of distribution of all states up to time t given only states up to time $t-1$) and the update step (the correction of the predicted distribution taking into account the new observation) as follows:

$$\begin{aligned} p(x_{0:t} \mid y_{0:t-1}) &= p(x_{0:t-1} \mid y_{0:t-1}) f(x_t \mid x_{t-1}) \\ p(x_{0:t} \mid y_{0:t}) &= \frac{p(x_{0:t} \mid y_{0:t-1}) g(y_t \mid x_t)}{p(y_t \mid y_{0:t-1})} \end{aligned} \tag{2.11}$$

Moreover, the estimate of any other smoothing distribution $p(x_{j:k} \mid y_{1:t})$ where $(0 \leq j \leq k \leq l \leq t)$ can be obtained by integrating out x_i that are not interested in as follows:

$$p(x_{j:l} \mid y_{1:t}) = \int p(x_{1:t} \mid y_{1:t}) dx_{1:j-1, l+1:t} \tag{2.12}$$

A common smoothing distribution of interest is the marginal distribution at time t, $p(x_t \mid y_{1:t})$, which is also referred to as the filtering distribution.

Another distribution of interest is the prediction distribution, that is the estimation of the distribution of any unseen *future* states using only all observations up to current

time. If we let $j = 1$ and $l \geq t$ in (2.12), we obtain the following equation:

$$p(x_{j:l} \mid y_{1:t}) = p(x_{j:t} \mid y_{1:t}) \prod_{i=t+1}^{l} f(x_i \mid x_{i-1}) \tag{2.13}$$

Therefore, any prediction density can be obtained by simply integrating out the variables of not interest from the above equation.

While the distribution estimation problem may appear to be simple, it is in fact far from being resolved in practice. The integral appear in the above equations are often intractable and can only be estimated except in the very specific setting as discussed below.

2.5.2 Kalman Filter

In the linear Gaussian setting in which the transition density and likelihood are Gaussian distributions with center lied at a point of a linear combination of the known conditional variables, u_t of the following form:

$$\begin{aligned} f_t(x_t \mid x_{t-1}, u_t) &= N(A_t(u_t)x_{t-1} + F_t(u_t), B_t(u_t)B_t(u_t)^T) \\ g_t(y_t \mid x_t, u_t) &= N(C_t(u_t)x_t, D_t(u_t)D_t(u_t)^T) \end{aligned} \tag{2.14}$$

where A_t, B_t, C_t, D_t are known matrix or vector operations, which of each may depend on the given conditional variable u_t.

Using the properties of Gaussian distribution, the integral involved can be resolved analytically. This leads to the widely used *Kalman Filter* [7] that has the following recursive solution as follows:

$$\mu_{t|t-1} = A_t(u_t)(\mu_{t-1|t-1})X_{t-1} + F_t(u_t) \tag{2.15}$$

$$\Sigma_{t|t-1} = A_t(u_t)\Sigma_{t-1|t-1}A_t(u_t)^T + B_t(u_t)B_t(u_t)^T \tag{2.16}$$

$$S_t = C_t(u_t)\Sigma_{t|t-1}C_t(u_t)^T + D_t(u_t)D_t(u_t)^T \tag{2.17}$$

$$y_{t|t-1} = C_t(u_t)\mu_{t|t-1} + G_t(u_t) \tag{2.18}$$

$$\mu_{t|t} = \mu_{t|t-1} + \Sigma_{t|t-1}C_t(u_t)S_t^{-1}(y_t - y_{t|t-1}) \tag{2.19}$$

$$\Sigma_{t|t} = \Sigma_{t|t-1} - \Sigma_{t|t-1}C_t(u_t)S_t^{-1}C_t(u_t)\Sigma_{t|t-1} \tag{2.20}$$

where $\mu_{t|t-1}$ and $\Sigma_{t|t-1}$ are the predicted mean and co-variance matrix of the state x_t, $y_{t|t-1}$ and S_t are the mean and co-variance matrix of the measurement at time t and $\mu_{t|t}$ and $\Sigma_{t|t}$ are the estimated mean and co-variance matrix of the state x_t after seeing the observation y_t.

There are various extensions have been developed on top of this approach. For example, the Extended Kalman Filter (EKF) which uses Taylor Series expansion to linearize at the conditional variables locally [8], Unscentend Kalman Filter which further extend the idea in EKF by only using a minimal set of well chosen samples [9], etc.

2.5.3 Sequential Important Sampling (SIS)

In more general setting, there is however no analytical solution for this estimation problem. Sequential Monte Carlo provides a systematic way to approximate the solution to this estimation problem. Assuming that it is possible to decompose the selected importance distribution into recursion form as follows:

$$\begin{aligned} q_{0:t}(x_{0:t} \mid y_{0:t}) &= q_{0:t-1}(x_{0:t-1} \mid y_{0:t-1}) q_t(x_t \mid x_{t-1}, y_t) \\ &= q_0(x_0 \mid y_0) \prod_{i=1}^{t} q_i(x_i \mid x_{i-1}, y_i) \end{aligned} \tag{2.21}$$

then it is possible to obtain sample $x_{0:t}$ by first sampling $x_0 \sim q_0(\cdot)$ at time 0 and then $x_i \sim q_i(x_i \mid x_{i-1}, y_i)$ for all time i from 1 to t. The corresponding weights associated to each sample $x_{0:t}$ can also be decomposed into recursion form as follows:

$$\begin{aligned} \tilde{w_t} &= \frac{p_{0:t}(x_{0:t} \mid y_{0:t})}{q_{0:t}(x_{0:t} \mid y_{0:t})} \\ &= \frac{p_{0:t-1}(x_{0:t-1} \mid y_{0:t-1})}{q_{0:t-1}(x_{0:t-1} \mid y_{0:t-1})} \frac{p_{0:t}(x_{0:t} \mid y_{0:t})}{p_{0:t-1}(x_{0:t-1} \mid y_{0:t-1}) q_t(x_t \mid x_{0:t-1}, y_t)} \\ &= \tilde{w}_{t-1} \frac{p_{0:t}(x_{0:t} \mid y_{0:t})}{p_{0:t-1}(x_{0:t-1} \mid y_{0:t-1}) q_t(x_t \mid x_{0:t-1}, y_t)} \\ &\propto \tilde{w}_{t-1} \frac{f_t(x_t \mid x_{t-1}) g_t(y_y \mid x_t)}{q_t(x_t \mid x_{t-1}, y_t)} \qquad (2.22) \\ &\propto \tilde{w}_0 \prod_{i=1}^{t} \alpha_i(x_i) \qquad (2.23) \end{aligned}$$

where $\alpha_t(x_t)$ is often referred to as incremental importance weight function. This is the key concept in SIS.

Using these weighted samples $\{(x_{0:t}^{(i)}, \tilde{w}_t^{(i)})\}_{1 \le i \le N}$, it is possible to estimate any function f defined on the space using the self normalised importance sampling estimator in the same way as (2.8) as follows:

$$\begin{aligned} \hat{f}(x_{0:t}) &= \sum_{i=1}^{N} \frac{\tilde{w}_t^{(i)}}{\sum_{j=1}^{N} \tilde{w}_t^{(j)}} f(x_{0:t}^{(i)}) \\ &= \sum_{i=1}^{N} \hat{w}^{(i)} f(x_{0:t}^{(i)}) \end{aligned} \tag{2.24}$$

The SIS is summarised in Algorithm 3.

Algorithm 3 Sequential Importance Sampling

function SequentialImportanceSampling(N, T)
  Set $t \leftarrow 0$.
  For $i \in 1, \ldots, N$, sample $x_0^{(i)} \sim q(x_0^{(i)} \mid y_0^{(i)})$.
  For $i \in 1, \ldots, N$, calculate the unnormalised importance weight:

$$\tilde{w}_0^{(i)} = \frac{p(x_0^{(i)}) g_0(y_0 \mid x_0^{(i)})}{q_0(x_0^{(i)})}$$

  For $i \in 1, \ldots, N$, normalize the importance weight:

$$\hat{w}_0^{(i)} = \frac{\tilde{w}_0^{(i)}}{\sum_{i=1}^{N} \tilde{w}_0^{(i)}}$$

  Set $t \leftarrow t + 1$.
  while $t \leq T$ **do**
    For $i \in 1, \ldots, N$, sample $x_t^{(i)} \sim q(x_t^{(i)} \mid y_{t-1}^{(i)}, x_{t-1}^{(i)})$.
    For $i \in 1, \ldots, N$, set $x_{0:t}^{(i)} \leftarrow (x_{0:t-1}^{(i)}, x_t^{(i)})$.
    For $i \in 1, \ldots, N$, calculate the unnormalised importance weight:

$$\tilde{w}_t^{(i)} = w_{t-1}^{(i)} \frac{f_t(x_t^{(i)} \mid x_{t-1}^{(i)}) g_t(y_t \mid x_t^{(i)})}{q_t(x_t^{(i)} \mid x_{t-1}^{(i)})}$$

    For $i \in 1, \ldots, N$, normalize the importance weight:

$$\hat{w}_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{i=1}^{N} \tilde{w}_t^{(i)}}$$

  end while
end function

2.5.4 Optimal Proposal Distribution

While SIS is attractive, it is nothing but a specialised version of importance sampling introduced earlier in Section 2.4. As the state space increases with the number of time step t, direct importance sampling on a state space that is increasing in size is not very efficient. The weights of the samples will degenerate over time, in the sense that the weights start to concentrate on only a small number of samples. Consequently, many samples will have negligible weights and do not contribute much in the estimating the expectation. See [10] for a step-by-step illustration. The weight degeneracy issue cause the quality of the estimator degrades over time.

To alleviate this issue, looking at (2.22), it is obvious that the variance of importance weight can be minimised by using a proposal distribution of the following form:

$$q_t(x_t \mid x_{t-1}, y_t) \propto f_t(x_t \mid x_{t-1}) g_t(y_t \mid x_t) \tag{2.25}$$

This is often referred to as the optimal proposal distribution.

In general, it is not always possible to sample from this optimal proposal distribution. Yet, the knowledge of its form can be helpful in designing a reasonable good proposal distribution, which one can sample from. Better proposal reduces the amount of variance introduced, but it does not eliminate the weight degeneracy problem in general.

2.5.5 Sequential Importance Resampling (SIR)

The variance in importance weight accumulates over iterations. This suggests a possible solution is to "reset" the weights associated to the samples somehow during the iterations. Sequential Importance Resampling (SIR) introduces an additional resampling step to SIS step in a similar fashsion as discussed in Section 2.4. After resampling, the weight of each samples are reset to be equal, i.e., $\frac{1}{N}$.

Besides the simpliest multinomial resampling scheme, many different resampling schemes have been proposed in the literature. For example, stratified resampling [11] as the name suggested splitting the samples into stratums to ensure the good coverage on the resulting sample set, residual resampling [12] which is able to decrease the variance of the weights due to resampling, etc. See [13] for further details on the comparison of these sampling schemes. The SIR algorithm is now summarised in Algorithm 4.

Algorithm 4 Sequential Importance Resampling

function SequentialImportanceResampling(N, T)
  Set $t \leftarrow 0$.
  For $i \in 1, \ldots, N$, sample $x_0^{(i)} \sim q(x_0^{(i)} \mid y_0^{(i)})$.
  For $i \in 1, \ldots, N$, calculate the unnormalised importance weight:

$$\tilde{w}_0^{(i)} = \frac{p(x_0^{(i)}) g_0(y_0 \mid x_0^{(i)})}{q_0(x_0^{(i)})}$$

  For $i \in 1, \ldots, N$, normalize the importance weight:

$$\hat{w}_0^{(i)} = \frac{\tilde{w}_0^{(i)}}{\sum_{i=1}^N \tilde{w}_0^{(i)}}$$

  Set $t \leftarrow t + 1$.
  while $t \leq T$ **do**
    For $i \in 1, \ldots, N$, sample $x_t^{(i)} \sim q(x_t^{(i)} \mid y_{t-1}^{(i)}, x_{t-1}^{(i)})$.
    For $i \in 1, \ldots, N$, set $x_{0:t}^{(i)} \leftarrow (x_{0:t-1}^{(i)}, x_t^{(i)})$.
    For $i \in 1, \ldots, N$, calculate the unnormalised importance weight:

$$\tilde{w}_t^{(i)} = w_{t-1}^{(i)} \frac{f_t(x_t^{(i)} \mid x_{t-1}^{(i)}) g_t(y_t \mid x_t^{(i)})}{q_t(x_t^{(i)} \mid x_{t-1}^{(i)})}$$

    For $i \in 1, \ldots, N$, normalize the importance weight:

$$\hat{w}_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{i=1}^N \tilde{w}_t^{(i)}}$$

    Resample: For $i \in 1, \ldots, N$, resample $x_t^{(i)} \sim \frac{\sum_{i=1}^N \hat{w}_t^{(i)} \delta_{\tilde{x}_{t,0:t}^{(i)}}}{\sum_{j=1}^N \hat{w}_t^{(j)}}$
  end while
end function

2.5.6 Effective sample size (ESS)

Resampling step induces additional Monte Carlo variance to the weights. Yet, this step is necessary to avoids accumulation of estimation variance on weights over time and therefore result in a more stable approximation to the filtering and preditive distribution.

To trading off these two competing requirements, one possible way to monitor the effective sample size (ESS) which provides a measure on the quality of the weighted samples which can be estimated as follows:

$$ESS \approx \frac{1}{E[w^2]} \approx \frac{\left(\sum_{i=0}^{N} w_i\right)^2}{\sum_{i=0}^{N} w_i^2} \tag{2.26}$$

A possible implementation is that resampling step is only triggered if the ESS_t fall below certain threshold at time t, say $\frac{N}{2}$. See [10] for detail discussion on ESS.

2.5.7 Resample-Move Algorithm

However, resampling is not a silver bullet for sampling imporishment. Essentially, resampling provides a mechanism to eliminate low weight samples to give way to replicate *copies* of high weight samples. This allows all samples to participate and contribute to the distribution estimation. This is obvious beneficial for the case of estimating filtering distribution and predictive distribution. Over time, this replication result in decrease in the number of distinct samples for previous time steps. Eventually, the start of the sample trajectory will become the same. This problem is known as sample impoverishment. This is a fundamental weakness of SMC, in which the sample path history is never re-written. This lose of diversity in the sample set will have a negative impact when it comes to estimating snoothing distribution.

To counteract this sample impoverishment, Resample-Move Algorithm [14] is proposed to introduce some perturbance to the samples (so to diversify them) without changing the distribution they represent. This is accomplished by using MCMC steps with a Markov Kernel, K that is invariant to the target distribution, $p(\cdot)$. In the original paper, this is achieved by simply introducing an additional MCMC "move" step to each sample after the resampling step using a kernel K that is invariant to the target distribution. This Resample-Move algorithm is summarised in Algorition 5.

This does not entirely solve the smoothing distribution estimation problem. To apply Markov Kernels with invariant distribution corresponding to the smoothing distribution, the space that Markov kernel is defined has to increase at each iteration. This implies the computation time increases linearly with time. Moreover, fast mixing

high dimension Markov kernel in itself is not easy to design. In practice, one could use a sliding windows approach, in which MCMC Kernels which diversify the samples of the previous n time step at each iteration. This has a *fixed* additional cost to each iteration.

Algorithm 5 Resample-Move Algorithm

function ResampleMoveAlgorithm(N, T)
  Set $t \leftarrow 0$.
  For $i \in 1, \ldots, N$, sample $x_0^{(i)} \sim q(x_0^{(i)} \mid y_0^{(i)})$.
  For $i \in 1, \ldots, N$, calculate the unnormalised importance weight:

$$\tilde{w}_0^{(i)} = \frac{p(x_0^{(i)}) g_0(y_0 \mid x_0^{(i)})}{q_0(x_0^{(i)})}$$

  For $i \in 1, \ldots, N$, normalize the importance weight:

$$\hat{w}_0^{(i)} = \frac{\tilde{w}_0^{(i)}}{\sum_{i=1}^N \tilde{w}_0^{(i)}}$$

  Set $t \leftarrow t + 1$.
  while $t \leq T$ **do**
    For $i \in 1, \ldots, N$, sample $x_t^{(i)} \sim q(x_t^{(i)} \mid y_{t-1}^{(i)}, x_{t-1}^{(i)})$.
    For $i \in 1, \ldots, N$, set $x_{0:t}^{(i)} \leftarrow (x_{0:t-1}^{(i)}, x_t^{(i)})$.
    For $i \in 1, \ldots, N$, calculate the unnormalised importance weight:

$$\tilde{w}_t^{(i)} = w_{t-1}^{(i)} \frac{f_t(x_t^{(i)} \mid x_{t-1}^{(i)}) g_t(y_t \mid x_t^{(i)})}{q_t(x_t^{(i)} \mid x_{t-1}^{(i)})}$$

    For $i \in 1, \ldots, N$, normalize the importance weight:

$$\hat{w}_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{i=1}^N \tilde{w}_t^{(i)}}$$

    Resample: For $i \in 1, \ldots, N$, resample $x_t^{(i)} \sim \frac{\sum_{i=1}^N \hat{w}_t^{(i)} \delta_{\tilde{x}_{t,0:t}^{(i)}}}{\sum_{j=1}^N \hat{w}_t^{(j)}}$
    Move: For $i \in 1, \ldots, N$, sample $x_{t,0:t}^{(i)} \sim K_t(\cdot)$, where K_t is p_t-invariant.
  end while
end function

2.5.8 Marginalised SMC

In practice, many models may not be entirely Gaussian. Some states may be linear and Gaussian, conditionally upon the other states. The naive way to model this is to use SMC to model all the states. A better approach would be making use of the Gaussian properties. The states are split into two categories: the linear Gaussian states and the non-linear states. Then, the Kalman Filter which is optimal and possess a closed form solution can be used to model the linear Gaussian states and the SMC can be to model the non-linear states. This marginalisation setting will yield estimates with smaller variances.

To illustrate this, let consider the a simple conditional linear Gaussian Model introduced earlier in Section 2.5.2 with the model dynamic spedicfied as follows:

$$p_t(u_t \mid u_{t-1}) = (\text{any given form}) \tag{2.27}$$

$$p_t(x_t \mid x_{t-1}, u_t) = N(A_t(u_t)x_{t-1} + F_t(u_t), B_t(u_t)B_t(u_t)^T)$$

$$p_t(y_t \mid x_t, u_t) = N(C_t(u_t)x_t, D_t(u_t)D_t(u_t)^T) \tag{2.28}$$

where A_t, B_t, C_t, D_t and F_t are known matrix or vector operations.

At any time t, the full posterior distribution of this model can be factorised as follows:

$$p(u_{0:t}, x_{0:t} \mid y_{0:t}) = p(x_{0:t} \mid u_{0:t}, y_{0:t})p(u_{0:t} \mid y_{0:t}) \tag{2.29}$$

Looking at the right hand side of the equation, the first term follows Gaussian distribution, and therefore can be computed optimally using Kalman Filter. For the second term, we can rewrite it into the following recursion form as follows:

$$\begin{aligned} p(u_{0:t} \mid y_{0:t}) &\propto p(y_k \mid u_{0:t}, y_{0:t-1})p(u_{0:t} \mid y_{0:t-1}) \\ &= p(y_k \mid u_{0:t}, y_{0:t-1})p(u_t \mid u_{0:t-1}, y_{0:t-1})p(u_{0:t-1} \mid y_{0:t-1}) \\ &= p(u_{0:t-1} \mid y_{0:t-1})p(y_k \mid u_{0:t}, y_{0:t-1})p(u_t \mid u_{t-1}, y_{0:t-1}) \end{aligned} \tag{2.30}$$

Assuming that it is possible to decompose the selected importance distribution into recursion form as follows:

$$\begin{aligned} q_{0:t}(x_{0:t} \mid y_{0:t}) &= q_{0:t-1}(x_{0:t-1} \mid y_{0:t-1})q_t(x_t \mid x_{t-1}, y_t) \\ q_{0:t}(u_{0:t} \mid y_{0:t}) &= q_{0:t-1}(u_{0:t-1} \mid y_{0:t-1})q_t(u_t \mid u_{0:t-1}, y_{0:t}) \end{aligned} \tag{2.31}$$

then the associated unnormalised weight of each sample can be derived in a similar fashion as given in Section2.5.3 into the recursion form as follows:

$$\tilde{w}_t = \tilde{w}_{t-1}\frac{p_t(u_t \mid u_{t-1})p_t(y_t \mid u_{0:t}, y_{0:t-1})}{q_t(u_t \mid u_{0:t-1}, y_{0:t})} \tag{2.32}$$

Compare to (2.21), the main difference here is that $p(u_t \mid u_{0:t-1}, y_{0:t-1})p(u_{0:t-1} \mid y_{0:t-1})$ depends on the whole path space from time 0 up to t. The full Marginalised SMC is summarised in Algorithm 6.

The discussion has been focused on linear Gaussian model. There is another important class model is the discrete state-space HMMs, in which the model states are discrete. In scuh a model, HMM forward algorithm [15] can be used to marginalise these discrete states. See [6] for further detail.

2.6 Conclusion

This chapter presents a review of Monte Carlo method, with a particular focus on Sequential Monte Carlo method that is used extensively in this thesis for portfolio optimisation. It begins to describe traditional Monte Carlo sampling techniques. It then introduce a simple SMC algorithm, and couple of extensions that have been proposed to improve the performance of the algorithm.

It is worth nothing SMC is not restricted to sequential filtering problem. For example, it has been established that it is possible to use SMC within MCMC framework (pMCMC, where p stands for particle) [16] to solve other problems. In the next chapter, we will show Sequential Monte Carlo is used as a maximiser to search for a optimal strategy for a portfolio, given a multiplicative reward function.

Algorithm 6 Marginalised Sequential Monte Carlo

function MarginalisedSequentialMonteCarlo(N, T)

Set $t \leftarrow 0$.

For $i \in 1, \ldots, N$, sample $x_0^{(i)} \sim q(x_0^{(i)} \mid y_0^{(i)})$.

For $i \in 1, \ldots, N$, calculate the unnormalised importance weight:

$$\tilde{w}_0^{(i)} = \frac{p(x_0^{(i)}) g_0(y_0 \mid x_0^{(i)})}{q_0(x_0^{(i)})}$$

For $i \in 1, \ldots, N$, normalize the importance weight:

$$\hat{w}_0^{(i)} = \frac{\tilde{w}_0^{(i)}}{\sum_{i=1}^{N} \tilde{w}_0^{(i)}}$$

Set $t \leftarrow t + 1$.

while $t \leq T$ **do**

For $i \in 1, \ldots, N$, sample $u_t^{(i)} \sim q(u_t^{(i)} \mid y_{t-1}^{(i)}, u_{t-1}^{(i)})$.

For $i \in 1, \ldots, N$, calculate the unnormalised importance weight:

$$\tilde{w}_t = \tilde{w}_{t-1} \frac{p_t(u_t \mid u_{t-1}) p_t(y_t \mid u_{0:t}, y_{0:t-1})}{q_t(u_t \mid u_{0:t-1}, y_{0:t})}$$

where

$$p(y_t \mid u_{0:t}^{(i)}, y_{1:t-1}) \sim N(y_{t|t-1}, S_t) \tag{2.33}$$

with $y_{t|t-1}$ and S_t are updated with Kalman Filter:

$$\begin{aligned}
\mu_{t|t-1} &= A_t(u_t)(\mu_{t-1|t-1}) X_{t-1} + F_t(u_t) \\
\Sigma_{t|t-1} &= A_t(u_t) \Sigma_{t-1|t-1} A_t(u_t)^T + B_t(u_t) B_t(u_t)^T \\
S_t &= C_t(u_t) \Sigma_{t|t-1} C_t(u_t)^T + D_t(u_t) D_t(u_t)^T \\
y_{t|t-1} &= C_t(u_t) \mu_{t|t-1} + G_t(u_t) \\
\mu_{t|t} &= \mu_{t|t-1} + \Sigma_{t|t-1} C_t(u_t) S_t^{-1} (y_t - y_{t|t-1}) \\
\Sigma_{t|t} &= \Sigma_{t|t-1} - \Sigma_{t|t-1} C_t(u_t) S_t^{-1} C_t(u_t) \Sigma_{t|t-1}
\end{aligned}$$

For $i \in 1, \ldots, N$, normalize the importance weight:

$$\hat{w}_t^{(i)} = \frac{\tilde{w}_t^{(i)}}{\sum_{i=1}^{N} \tilde{w}_t^{(i)}}$$

Resample: For $i \in 1, \ldots, N$, resample $x_t^{(i)} \sim \frac{\sum_{i=1}^{N} \hat{w}_t^{(i)} \delta_{\tilde{x}_{t,0:t}^{(i)}}}{\sum_{j=1}^{N} \hat{w}_t^{(j)}}$

end while

end function

Chapter 3

Portfolio optimisation

Optimal portfolio is the ultimate goal of every investment manager; but the optimality criterion can be very different to each of them. In the infamous Markowitz's modern portfolio theory [1]. it is assumed that investor attempt to maximize a portfolio's return and minimize the risk (measured by the variance of the portfolio return. For index tracker fund manager, the main objective of portfolio management is to track and replicate the exposure of a benchmark index[1]. It is the later where the focus of the thesis lie.

In this chapter, it begins with a brief introduction on index tracking fund, what it actually is and why it is becoming increasing important in the last decade and short summary on some of the causing factor of tracking error. It then begins to describe how this problem can be formulated as a stochastic control problem, which is then turned into a parameter estimation in SMC framework. It then presents some of the experiments carried out to verify this approach, starting with a simple proof of concept experiment in Section **??**, continue on with a more complicated example with real-world data in Section **??** and concluding the chapter with an experiment in demonstrating how the Model Predictive Control (MPC) can be incorporating into the basic framework.

3.1 Index Tracking Fund

An index can be thought of a summary statistics of a market, typically formed as weighted average of some financial instruments. For example, FTSE 1010 is an index attempts to represents the performance of the 100 biggest companies in UK. This index is however not a product one can invest directly.

[1]The lack of active management generally makes the fund less vulnerable to change of management and has the advatanges of lower fees and taxes

To allows investors to take exposure to the markets (as represented by the indices), index tracker funds are introduced. The funds generally follow very strict and transparent rules with the main objective to track their benchmark indices as close as possible. Investors of these funds therefore are only exposed to the market risks of the chosen indices, carving away most risk associated with the investment process of the traditional active fund management.

3.1.1 Index Replication

To track an index, the simplest way is to replicate the index by investing all the components of the benchmark index. However, this can be costly and different to manage, considering some of the indices may have huge number of components, e.g., MSCI World index that consists of 1612 components from many different countries. Instead, one could partial replicate the index by sampling some of the components that are most representative of hte index. This can mean those components with larger weights, more volatiles or less correlated in return (assuming that correlated components would have their returns "cancelled out" one another). This partial replication could save transaction cost, at the time same, introducing some tracking errors to the funds.

Instead of physically replicating the index by investing in the components, the portfolio manager could decide in entering a swap agreement with a counterparty (typically investment bank) that provides the exact return of the stock market or commodity its tracking. Essentially, this transfers the market risk to the counterparty, at the same introducing the counterparty default risk. This is known synthetic replication.

3.1.2 Causing factors of tracking error

To evaluate the performance of index tracking fund, different metrics have been introduced to quantify the mismatch between the performance of a fund and its benchmark. For example, tracking difference is the sum of absolute difference in returns between of a fund and its benchmark. Here, we adopt the tracking error as our metric, which is defined to be the standard deviation of the absolute difference of the returns of the fund and the benchmark defined in [17]. Formally, tracking error is defined to be:

$$\epsilon = \sqrt{\mathbb{E}[(r_p - r_b)^2]} \tag{3.1}$$

The tracking error can be caused by different factors: some of which can be summarised as follows:

1. Benchmark index rebalance — the benchmark index is re-weighting its constitutents periodically to reflex the changes on the market based on its methodology. To track the index, the fund has to adjust its portfolio accordingly. This will incur some trasaction costs. During the index rebalance period, cash drag can happen between the liquidation of the constituents that have weights reduced/dropped and the addition of the constituents that have weights increased/added. This cash is essentially not participating in the market and therefore do not reflex the changes on the benchmark index.

2. different assumption on dividend trinvestment and taxation — This is best illustrate with examples. For example, the benchmark index calculation may assume immediate reinvestment of dividends on ex-dividend dates but the fund may only able to reinvest the dividend after receiving it. The tax treatment may on the dividends amy also different too.

3. Replication and Sampling techniques — funds may choose too replicate the benchmakr index by selecting a subset of the constituents (often the ones with larger weights and more liquid) in an effort to minimize the transaction costs. This exclusion of the smaller, less liquid constitutents may introduce another source od tracking error, especially under stressed market.

4. Total Expense Ratio — the average annual expense that is charged to the fund on daily basis to cover the management cost.

This list is by no mean exclusive. See [17] for further detail.

3.2 Technical Approach

To carry out portfolio optimisation, various approaches have been explored in the past, mainly in an anlytical fahsion, in adopting necessary assumption as necessary. This seems rather restrictive; there are many instances where numerical method has been used to derive an approximate or even more effective solution to the problem in question. For example, Monte Carlo technique is used to do integral, evolutionary techniques applied in engineering domains, etc.

Our approach to the problem in this thesis is a radical one. We view a minimising portfolio tracking error as *stochastic* control problem. Instead of using a traditional deterministic state space models, we use as stochastic modelling approach to carry out the portfolio optimisation. In particular, we focus here the the problem in minimizing

the tracking error between a portfolio and its benchmark index using a stochastic control modelling approach. Our aim is to determine what investment actions (buy or sell) on the necessary constituents a portfolio manager has to do on a daily basis across the investment horizon to minimize the tracking error of the fund managed. We will proceed by presenting the stochastic state space model that we assume throughout this thesis. This model is by no mean to compete the state of the art model in realistic portfolio optimisation, but to motivate further work in this direction.

3.2.1 Model specification and optimisation objective

We adopt here the conditional linear Gaussian model, in which the transition density and likelihood are Gaussian distributions with center lied at a point of a linear combination of the known conditional control parameters, u_t of the following: g form:

$$\begin{aligned} f_t(x_t \mid x_{t-1}, u_t) &= N(A_t(u_t)x_{t-1} + F_t(u_t), B_t(u_t)B_t(u_t)^T) \\ g_t(y_t \mid x_t, u_t) &= N(C_t(u_t)x_t, D_t(u_t)D_t(u_t)^T) \end{aligned} \tag{3.2}$$

where A_t, B_t, C_t, D_t are known matrix or vector operations and $u_{t t\geq 0}$ is a determinstic control input sequence that is used regulate the hidden states.

With this model, the objective is to search for a sequence of controls $u_{1:t}$ that would result in a sequence of observations $y_{1:t}$ is closed to a reference signal $y^{ref}_{1:T}$. This problem is often known as stochastic regulation problem. We adopt here the following finite horizon multiplicative reward function:

$$J(u_{1:T}, y^{ref}_{1:T}, x_0) = \mathbb{E}_{x_0}\left[-\frac{1}{2}\sum_{t=1}^{T}\left(||y^{ref}_t C_t(u_t)x_t - G_t(u_t)||^2_{(D_t(u_t)D_t(u_t)^T)^{-1}} + ||u_t - \Xi_t u_{t-1}||^2_{L_t}\right)\right] \tag{3.3}$$

where the expectation is taken with respect to the whole path of the Markov Chain $X^T_{t t=1}$,, starting with $X_0 = x_0$, i.e., $E_{x_0}[\phi(X_{1:t})] = \int \phi(X_{1:T}) \prod f_t(x_t \mid x_{t-1})\, dx_{1:t}$, with Q_t, L_t and Ξ_t are assumed to be known. The corresponding optimal open loop policy is:

$$u^*_{1:T} = \arg\max_{u_{1:T}} J(u_{1:T}; y^{ref}_{1:t}; x_0) \tag{3.4}$$

assuming the maximum is indeed attainable and well defined. This reward resembles closely the risk sensitive control formulation described in []. However, the problem here has a slightly different structure due to the actual distribution used in the expectation. The absence of an explicit risk sensitivy constant and the presece of y^{ref} as an explicity set reference rathr than an actual observation of th state. Neverthe less, a risk sensitivty constant could still be added through D_n, L_n if necessary.

3.2.2 Problem formulation

Under the realm of Bayesian inference framework, we treat the control inputs u as random variables that admit a prior distribution. We will further assume the sequence of control u_n is a Markov process with transition distribution $f_t(u_i \mid u_{i-1}$. The objective is to compute the marginal posterior distribution density $p(u_{0:t} \mid y_{0:t}^{ref}$. This posterior desntiy function can be derived as follows:

$$p(u_{0:t} \mid y_{0:t}^{ref}) \propto p(u_{0:t})p(y_{0:t}^{ref} \mid u_{0:t}) \tag{3.5}$$

$$= \prod_{i=1}^{t} p(y^{ref} \mid y_{0:i-1}^{ref}, u_{0:i}) f(u_i \mid u_{i-1}) \tag{3.6}$$

$$\begin{aligned} p(u_{0:t} \mid y_{0:t}) &\propto p(y_k \mid u_{0:t}, y_{0:t-1}) p(u_{0:t} \mid y_{0:t-1}) \\ &= p(y_k \mid u_{0:t}, y_{0:t-1}) p(u_t \mid u_{0:t-1}, y_{0:t-1}) p(u_{0:t-1} \mid y_{0:t-1}) \\ &= p(u_{0:t-1} \mid y_{0:t-1}) p(y_k \mid u_{0:t}, y_{0:t-1}) p(u_t \mid u_{t-1}, y_{0:t-1}) \end{aligned} \tag{3.7}$$

We can solve this equation using SMC by consider the pair etc.

However, a more efficient algorithm can be derived by considering the following factorisation:

$$p(x_{0:t}, u_{0:t} \mid y_{1:t}) = p(x_{0:t} \mid u_{0:t}, y_{1:t}) p(u_{0:t} \mid y_{1:t}) \tag{3.8}$$

Note that density $P(x_{0:t} \mid u_{0:t}, y_{1:t}$ is Gaussian mixture model, which can be computed analytically usign Kalman Filfter given the density $p(u_{0:t} \mid y_{1:t}$, which has the following recursion form:

$$1 + 1 \tag{3.9}$$

This can be resolved using SMC by using particles to do this.

Assume for the time being that the control inputs are set as U1:n = u1:n and remain fixed. Recall Xnn0 and Ynn1 are assumed to be stochastic processes obeying a Markov transition density fn(xn—xn1,un) and a conditionally independent likelihood density gn(yn—xn,un) respectively. Given any observed y1:n realisation, inference about the states X1:n may be based on the following posterior density

show the algorithm

3.3 Numerical example on stationary oscillating wave

We will consider a simple linear Gaussian state space model as presented earlier as (**??**), with $A_t = B_t = C_t = D_t = I$, $F_t u_t = u_t$, $G_t u_t = 0$, $Xi = 0$, $u = 0$. This model

can be re-written as follows:

$$X_t = X_{t-1} + W_t + U_t, W_t \sim N(0, I) \tag{3.10}$$

$$Y_t = X_{t-1} + V_t, N_t \sim N(0, I) \tag{3.11}$$

with the target reference is set to be an oscillating wave: $y_t = \cos(0.2\pi t + 0.3)$. This toy example is first introduced in [18]. It serves two purposes here. Firstly, it provides a simple example to verify our implementation[1] Secondly, it severs as benchmark for the following experiments in which we attempt to use more complicated reference signals and models.

Setting the maximum time period, $T = 50$,

We proceed by examining the algorithm for the following different implementations: (a) qn = fn without using the MCMC move (Step 2(d)), (b) qn = fn with the MCMC move and (c) qn being the optimal importance density of [13] without the MCMC case. In the last case the MCMC step was omitted because when the optimal importance density is used the improvement in performance was marginal. In the MCMC move we will use a random walk proposal. For = 100, 1000 and N = 200, 500, 1000, 5000, 10000 we present box plots for logT(UI1:T)inFigure1after30independentrunsofthealgorithm,whereUI1:T istheestimatorofu1:T ineach run. Similarly, in Figure 2 we plot U I1:T and the particle population nU i1:T oNi=1 taken from one run of the each of the same cases, but this time we show results only for N = 10000. Simulations took roughly 3, 70 and 4 seconds per 1000 particles for (a), (b), (c) respectively when implemented in Matlab using a 2.4 GHz processor. The algorithm seems to perform quite well in most settings and very well when the optimal importance distribution is used. For the case where qn = fn, MCMC seems to improve the performance of the algorithm. The improvement is more evident when = 1000 both in the box plots and when plotting nUi1:T oNi=1, for which the degeneracy is apparent without the MCMC step.

3.3.1 Period length performance

We extend the time step to be 90 and 250 steps and looks at the corresponding performance. Due to the increase of time step, it makes sense to have more particles to track them and also look at the performance of performance of different order of γ. Based on the same metric, the results are summarised in Figure X.

[1] Strictly speaking, testing increases confidence but does not prove no bug, which is almost impossible in practice.

3.3.2 Increasing the power

3.4 Different reference signals

Given the initial result looks promising, we attempt to investigate with the following more complicated reference signals:

1. reference signal treding osccilating wave —
2. two un-correlated bi-variate signals —
3. two correlated bi-variate signals —
4. ten different signals —

3.5 Discussion and possible extension

Chapter 4

Evaluation and conclusions

The work reported in previous chapters provides evidence to support the thesis hypothesis stated in Section, namely:

> Evolutionary algorithms (EAs) have the potential to be an effective means of determining the security policies that suit dynamic challenging environments.

This chapter reviews the work that has been done, evaluates the extent to which they justify the thesis hypothesis and concludes the thesis by addressing the directions for future work.

4.1 Evaluation

In previous three chapters, We have detailed several experimentations that serves to support the thesis hypothesis from three different strands of research. We explored the potential of EAs in inferring optimal security policies, dynamically updating security policies with new decision examples and searching for policies with optimal trade offs between objectives using simulation runs. This section summarises the work completed in each strand of research and outlines the contributions and novelty of the work presented in this thesis.

4.1.1 Static policy inference

Current security policy is often developed in a top-down approach. High-level security goals are first determined, after which they undergo a series of refinement processes to obtain the low-level executable rules. Although some work has been done in applying machine learning techniques to aid the policy refinement process, there is no previous

work to my knowledge in the application of EAs or machine learning techniques in inferring security policies.

Chapter details the experiments in using EAs to infer security policies from decision examples. Here EAs is used as a tool to generalise a set of low-level examples to a set of high-level rules. Various simple security policies have been attempted and inferred successfully. These include the traditional MLS Bell-LaPadula policy model, the budgetised MLS policy model and the Fuzzy MLS policy model. Two different EAs, namely GP and GE are used. In all cases, the results show that a minimal amount of design effort and domain knowledge are required to infer the reference policy or a close approximator of it. The only requirements are to have a good fitness function and training examples that form a good representation of the target policy.

The last part of the chapter presents how other machine techniques can be incorporated into the policy inference framework created. Fuzzy set concept is used as an example here. Multiple policies are learnt independently; each of which focuses on inferring a fuzzy rule for a particular class of decisions (fuzzification). The ultimate output policy, which is an ensemble of all these policies, is formed using a weighted voting mechanism (defuzzification). Various experiments have been carried out to examine different fuzzification and defuzzification techniques. The results show that these approaches can consistently infer policies that closely match with the original reference models used.

4.1.2 Dynamic policy inference

There will inevitably be times when unseen circumstances demand a decision during operation. In some cases the default automated response may be imperative; in other cases this may be ill-advised. Manual decisions made to override the default one essentially define a new policy. Furthermore, even if the optimal security policy can be developed or inferred automatically, it would eventually become suboptimal due to the changes in either the operational environment or security requirements, or both. Therefore, a security policy has to be able to continually change and be updated to suit the operational needs to maintain its optimality.

on dynamic security policy inference. As there is no dynamic security policy model available and therefore no decision example is available for us to work with, we designed a dynamic security policy model. This model is used to generate time varying decision examples for training and evaluation purposes.

To infer this dynamic security policy model, two novel dynamic learning frameworks based upon MOEAs are designed: one that based on Fan's intuition [19] and DOO.

In DOO, an n-objective optimisation problem is treated as a $2n$-objective optimisation problem by adding an opposing objective for each of the original objectives. With such a setting, DOO is able to maintain the diversity among the individuals in the population whilst optimising the intended objectives. This diversity can aid in preventing the population from premature convergence and allows the concept drift in the policy to be continually relearnt. The results show that these frameworks are very promising. Reasonably good approximators to the model are able to be inferred from the examples using these frameworks.

4.1.3 Mission-specific policy discovery

Chapter introduces the notion of mission-specific policy discovery. EAs are used to search for the security policies that can provide the optimal, or at least excellent, tradeoffs among security objectives for a specific mission. Here, EAs serve as an optimisation tool to synthesise the optimal policies, in terms of achieving the mission as well as security objectives without violating the constraints given.

We demonstrate here how simulation can be used to obtain the fitnesses of the policy candidates that are used to guide the policy search. To evaluate the fitness of an individual (policy) for a mission, the policy is first plugged into a simulated mission, then the simulated mission is executed and the outcome of it is measured. This is very different from the practice of fitting a policy a priori without the details of the specific mission being taken into account. This concept of "mission-specific policy" is entirely novel.

Various EA baed techniques are used here to discover the optimal policies. These include GP/MOGP and DOO. In all cases, the results show that these techniques are able to discover the set of policies that are optimal for the mission of concern. By using MOGP (and MOEAs in general), tradeoffs between a variety of criteria can be explored. Such information can be valuable to policy makers to select and apply the optimal policy that best fits the current operation.

4.1.4 Thesis contributions

In summary, we demonstrate how:

- EAs can be used to infer static security policies from a set of decision examples. Three different ways of representing security policies and two different EAs are investigated. The results show that this idea is feasible.

- the fuzzy set concept can be integrated into the policy inference framework to improve the policy inference performance. The idea is sufficiently generic to be applied to other classification problems, provided that there is a partial ordering among the classes.

- multi-objective evolutionary algorithms (MOEAs) can be used to infer dynamic security policies from a set of decision examples. Two novel dynamic learning frameworks based upon MOEAs are developed: one that is based on Fan's intuition and DOO. Both of them can be used as general dynamic classification algorithms.

- an ensemble policy model can be constructed from multiple models in a single EA run to achieve better performance. The improvement is especially significant in the DOO setting.

- MOEAs can be used to infer a set of Pareto optimal policies that fit a specific mission (or at least a specific family of missions).

- simulation runs can be used in place of a set of decision examples to provide feedback in evaluating the fitness of a policy with respect to the specified high-level objectives.

- MOEAs can be used as a decision making tool where tradeoffs between objectives exists. The Pareto front of the security policies discovered using MOEAs can reveal useful information about the relationship among different objectives, which may be difficult to obtain otherwise. Such information provides useful insight for policy makers to select and apply the optimal policy that fits the needs of current operational environment on a case-by-case basis.

4.2 Envisaged future work

Having discussed the contributions of the thesis, we now outline numerous possible directions for future work that have been identified during the course of this research.

4.2.1 Policy fusion

In dynamic coalitions, parties with different policies can come together to collaborate. Prior to the formation of dynamic coalitions, each party may have its own security policy. An interesting step forward would be to investigate how well EAs could be used to combine these security policies together. One possible way is to generate decision

examples from both existing policies and use these examples as the training input for the policy inference framework. MOEAs can also be used to discover the Pareto optimal set of policy candidates, which are then chosen depending on the security requirements. However there are still issues that require further investigation. These include:

- Understanding how to deal with policies that consists of different sets of decision-making factors, which may be measured using different scales.
- Understanding what the implicit priorities that EAs have assigned to the conflicting rules are, what the factors that influence the priorities are and how to control these priorities, etc.

4.2.2 The robustness of a security policy

The framework proposed in this thesis has been shown to be effective in dynamically inferring the optimal policy. However, the optimality of a policy is not always the only factor of concern; the robustness in performance of a security policy in different environments may be equally important. This is especially so in a pervasive operating environment where the deployment of a new policy can be a difficult or expensive process. To incorporate this factor into the proposed framework, a way to quantify the robustness in performance of a security policy is required.

This measure also provides a way to determine the invariant part of the optimal policies for different operational environments of concern. The determination of this invariant part is doubly useful: Firstly, it can serve as a template or testing target in the policy development process. Secondly, it can help to protect the security policy inference framework from poisoning attack, which attempts to mislead the inference process in the favour of the attacker by the injection of specially crafted decision examples.

4.2.3 Scalability with the training set size

Scalability is a subtle issue. We have addressed some aspects of this issue. For example, we have shown the method scales well with the size of the training set. In the experiments presented in Chapter we have increased the size of the training set from 100 examples to 1000 examples and the results still remain consistent. Obviously, the fitness evaluation time would increase; 1000 examples take ten times longer than 100 examples to evaluate. This is unlike to be an issue in practice as the fitness evaluation of each individual can be executed in parallel if necessary. In Chapterwe have shown that DOO is able to evolve and update policies with decision examples in an incremental manner. However, there are still some issues remaining with these frameworks that

need to be investigated. This includes searching for appropriate techniques to sample old decision examples and examining the generality of the DOO framework.

4.2.4 More complex security policies

The security policies used in this thesis are rather simple. This can be potentially an issue. However, note that these policies are either real-world policies or proposals from major research institutes for real world use. They are simple, but by no means "toy" policies. Ultimately, we should strive for simple policies wherever is possible, but at the same time, we should also need to acknowledge that MANET policies may need legitimately to be much more complicated. To cope with complexity, instead of attempting to extract and discover the policies as a whole, we could simply target the areas that we need help. Humans produce security policies sequentially too, i.e., they consider in turn authentication policy, file access control, audit policy, etc. In practice, it is also often that there are some rules of thumbs and constraints that are dictated from on high. We do not need to extract these bits of a policy. Yet, there is still much to answer here, for example:

- Can EAs be used to evolve more complex policies or policies of other types, e.g., obligation policies? If not, how can we divide the security policies into smaller components in a systematic manner?
- How to incorporate the constraints imposed from on high into the policy inference framework to form a continuous learning loop in an efficient manner? Should we take such constraints into consideration in the evolution process? If so, how?

4.2.5 More complex scenarios

The scenario used in Chapter is relatively simple. It has only one type of agent in each team and one type of information. A real test of this approach would be to embed it within a more realistic simulated scenario, with more sophisticated information types, and realistic consequence models. Note that the simulated scenario may be much more complex but we are really interested in some of the measurable properties, which may only be few. For example, how many properties would an operational commander be interested in feasibly ready trading off? The techniques proposed here should be able to scale well with it.

4.3 Closing remarks

The work reported in this thesis demonstrates a considerable degree of originality supported by extensive experimentation. The case studies are necessarily limited given the limited amount of time frame. However, the results demonstrate that portfolio optimisation approaches using Sequential Monte Carlo techniques have very considerable promise. Everyone accepts that portfolio optimisation is difficult, and things are to worsen as the marketfinancial become more complex environments with increasing sophistication and subtlety of decision-making process. We recommend these approaches to the research community for further investigation.

References

[1] Harry Markowitz. Portfolio selection. *The journal of finance*, 7(1):77–91, 1952. 1, 20

[2] R. Tyrrell Rockafellar and Stanislav Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–41, 2000. 1

[3] Markus Fischer and Giulia Livieri. Continuous time mean-variance portfolio optimization through the mean field approach. 2014. 2

[4] Christian P. Robert and George Casella. *Monte Carlo Statistical Methods (Springer Texts in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. 5

[5] Arnaud Doucet and Adam M Johansen. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of Nonlinear Filtering*, 12:656–704, 2009. 5

[6] Olivier Cappé, Eric Moulines, and Tobias Ryden. *Inference in Hidden Markov Models (Springer Series in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005. 7, 8, 18

[7] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960. 10

[8] Greg Welch and Gary Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 1995. 11

[9] Eric A. Wan and Rudolph Van Der Merwe. The unscented kalman filter. In *Kalman Filtering and Neural Networks*, pages 221–280. Wiley, 2001. 11

[10] Adam M Johansen, Ludger Evers, and N Whiteley. Monte carlo methods. *Lecture Notes*, 200, 2010. 13, 15

[11] Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of Computational and Graphical Statistics*, 5(1):1–25, 1996. 13

[12] Jun S. Liu and Rong Chen. Sequential monte carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93:1032–1044, 1998. 13

[13] R. Douc and O. Cappe. Comparison of resampling schemes for particle filtering. In *Image and Signal Processing and Analysis, 2005. ISPA 2005. Proceedings of the 4th International Symposium on*, pages 64–69, Sept 2005. 13

[14] Carlo Berzuini and Walter Gilks. Resample-move filtering with cross-model jumps. In Arnaud Doucet, Nando de Freitas, and Neil Gordon, editors, *Sequential Monte Carlo Methods in Practice*, Statistics for Engineering and Information Science, pages 117–138. Springer New York, 2001. 15

[15] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *PROCEEDINGS OF THE IEEE*, pages 257–286, 1989. 18

[16] Christophe Andrieu, Arnaud Doucet, and Roman Holenstein. Particle Markov chain Monte Carlo methods. *Journal of the Royal Statistical Society Series B*, 72(3):269–342, 2010. 18

[17] Ben Johnson, Hortense Bioy, Alastair Kellett, and Lee Davidson. On the right track: Measuring tracking efficiency in etfs. February 2013. 21, 22

[18] P. Del Moral N. Kantas and R. Vinter. Particle methods for stochastic regulation: towards an application for power scheduling. Technical report, Department of Electrical and Electronic Engineering, Imperial Colege, London, March 2011. 24

[19] Wei Fan. Systematic data selection to mine concept-drifting data streams. In *KDD '04: Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 128–137, New York, NY, USA, 2004. ACM. 28