Efficient Gaussian Process Based Optimistic Knapsack Sampling with Applications to Stochastic Resource Allocation

Author
Sondre Glimsdal

Supervisor
Ole-Christoffer Granmo

This Masters Thesis is carried out as a part of the education at the University of Agder and is therefore approved as a part of this education. However, this does not imply that the University answers for the methods that are used or the conclusions that are drawn.

University of Agder
Faculty of Engineering and Science
Department of ICT, Jun 3, 2013

Abstract

The stochastic non-linear fractional knapsack problem is a challenging optimization problem with numerous applications, including resource allocation. The goal is to find the most valuable mix of materials that fits within a knapsack of fixed capacity. When the value functions of the involved materials are fully known and differentiable, the most valuable mixture can be found by direct application of Lagrange multipliers. However, in many real-world applications, such as web polling, information about material value is uncertain, and in many cases missing altogether. Surprisingly, without prior information about material value, the recently proposed Learning Automata Knapsack Game (LAKG) and Hierarchy of Twofold Resource Allocation Automata (H-TRAA) offers arbitrarily accurate convergence towards the optimal solution, simply by interacting with the knapsack on-line. This paper introduces Gaussian Process based Optimistic Knapsack Sampling (GPOKS) a novel model-based reinforcement learning scheme for solving stochastic fractional knapsack problems, founded on Gaussian Process (GP) enabled Optimistic Thompson Sampling (OTS). Not only does this scheme converge significantly faster than LAKG, GPOKS also incorporates GP based learning of the material values themselves, forming the basis for OTS supported balancing between exploration and exploitation. Using resource allocation in web polling as a proof-of-concept application, our empirical results show that GPOKS consistently outperforms LAKG and H-TRAA, the current top-performers, under a wide variety of parameter settings.

Keywords: *Gaussian Process*, *Thompson Sampling*, *SNEFKS Problem Stochastic Resource Allocation Web Polling*

Contents

1 Introduction 7

1.1 Background and Motivation 7

1.2 Application: Web-polling 8

1.3 Thesis definition 9

1.4 Summary of Contributions 9

1.5 Outline 10

2 Knapsack Problems 10

2.1 Traditional Knapsack problems 10

2.2 The Nonlinear Equality FK (NEFK) Problem 12

2.3 The Stochastic Nonlinear Equality Knapsack (SNEFK) Problem 14

2.4 Web-Polling as a Stochastic NEFK problem 15

3 Reinforcement Learning 16

3.1 Learning Automata 17

3.2 Thompson Sampling 18

3.2.1 Multi Arm Bandit Problem 18

3.2.2 Applications of MABP 20

3.3 Thompson Sampling in a Bandit Context 20

3.3.1 TS Algorithm 21

3.3.2 Optimistic Thompson Sampling 22

4 Gaussian Processes 22

4.1 Machine learning as a Function Ensemble 23

4.2 Bayesian Linear Regression Model 25

4.3 Kernel-Induced Feature Spaces 28

4.4 Gaussian Process 29

4.4.1 Squared Exponential Kernel 30

4.4.2 GP Inference and Function Sampling 30

5 SFNEKS Problem - State-of-the-art solvers 32

5.1 LAKG – Learning Automata Knapsack Game 32

5.2 H-TRAA – Hierarchy of Twofold Resource Allocation Automata 33

5.3 GP-UCB – Gaussian Process Upper Confidence Bounds 34

5.3.1 Applying a MAB solver to the Stochastic NEFK 34

6 Gaussian Process based Optimistic Knapsack Sampling (GPOKS) 36

6.1 Sampling and solving a NEFK problem 36

6.1.1 Solving NEFK 37

6.1.2 Rejection Sampling 38

6.1.3 Optimistic Rejection Sampling 38

6.1.4 Monotonic Rejection Sampling 39

6.1.5 Rejection Sampling in GPOKS 40

6.2 Implementation 41

6.3 Example Steps 42
6.3.1 Finite Index Set Optimization 43

7 Experiment Configurations 45
7.1 GPOKS Variations 45
7.1.1 GPOKS Variants 45
7.2 Contending Algorithms 46
7.2.1 Learning Automata based Solvers 46
7.2.2 MAB Solvers 46
7.2.3 Naive solvers 47
7.2.4 Multiple sites 47

8 Experiment Results 48
8.1 Two Web Resources 48
8.1.1 Effective GPOKS 48
8.1.2 GPOKS Rejection Sampling 49
8.1.3 GPOKS Variations 50
8.1.4 GPOKS Contenders 52
8.2 Multiple Web Sites 53

9 Conclusions and Future Work 55

A DET-KS Proof 60

B Short Paper 61

Mathematical Notation

$\mathbf{x}$	vector $\mathbf{x}$
$\mathbf{x}'$	vector $\mathbf{x}$ transposed
$\langle \mathbf{x} \cdot \mathbf{z} \rangle$	The inner product between vector $\mathbf{x}$ and $\mathbf{z}$
$\lvert\mathbf{x}\rvert$	Euclidean length of vector $\mathbf{x}$
M	matrix M
I	Identity matrix of appropriate size
$\mathbb{E}[\cdot]$	Expected value operator
$P(\cdot)$	Probability operator
$\mathcal{N}(\mu, \sigma^2)$	Gaussian Distribution
$\mathcal{N}(\mu, \Sigma)$	Multivariate Gaussian Distribution
$\binom{n}{k}$	The binomial coefficient i.e. n choose k.
δ_{pq}	The Kronecker delta: $\delta_{pq} = 1$ if $p = q$, 0 otherwise.

List of Figures

1.1 P depends on both A and B. 7

2.1 NEFK Example 13

2.2 Time-line for two web-resources, a 'x' denotes a update for that resource. 15

3.1 Reinforcement Learning 17

3.2 Transition function $\mathcal{F}[\phi(t), \beta(t)]$ 18

3.3 MABP: Action a_2 is selected in time t and given its reward, there are 5 available arms. 19

3.4 A Gaussian distribution prior and posterior 21

3.5 TS and OTS, the potential sampling area is marked yellow. . . . 23

4.1 Ensemble adapting example step 2 and 4 25

4.2 Ensemble adapting example step 5 and 6 25

4.3 Bayesian Bivariate Regression, the yellow area marks a 95% confidence interval surrounding the mean of the predictive model. . . 28

4.4 SE-Kernel covariance between fixed point $x_p = 1.0$ and x_q. 30

4.5 GP Prior distribution, the marked yellow area is a 95% confidence interval. 31

4.6 GP Posterior after 1 observation, the marked yellow area is a 95% confidence interval. 31

4.7 GP Posterior after 2 observations, the marked yellow area is a 95% confidence interval. 32

4.8 GP Posterior after 4 observation, the marked yellow area is a 95% confidence interval. 32

5.1 H-TRAA with 8 materials (A - H) and some example weights. . 33

5.2 Number of arms when using MAB to solve SFNEKS 35

6.1 Conceptual overview of the different states of GPOKS; (1) Samples the GP based model for a similar deterministic NEFK problem. (2) Solves the NEFK problem. (3) Apply the solution to the NEFK problem to the original SNEFK problem. (4) Update the GP model to more closely represent the SNEFK problem. So that the next sample will be closer to the original problem. . . . 36

6.2 Samples from a GP without rejection sampling. 38

6.3 Samples from a GP using rejection sampling with a optimistic criterion. 39

6.4 Samples from a GP using rejection sampling with a monotonic criterion. 40

6.5 GPOKS Rejection Sampling with both monotonic and optimistic criteria. 41

6.6 GPOKS Architectural Overview 41

6.7 GP_1 and GP_2 at $t = 7$ 42

6.8 Estimate of material unit values $f_1'(x_1)$ and $f_2'(x_2)$ after 193 observations, with optimal and estimated material amounts x_1 and x_2. 43

7.1 Zipf distribution with $\alpha = 0.9$ and $\beta = 1.5$ 47

7.2 The test environments 48
8.1 Average running time for a single trial using FAST-GPOKS and REGULAR-GPOKS. Here $p_1/p_2 = 0.75/0.25$ 49
8.2 Rejection Sampling in GPOKS for the two material case 49
8.3 Variance in GPOKS Rejection Sampling Schemes for two materials, $p_1/p_2 = 0.75/0.25$ 50
8.4 White noise sensitivity in GPOKS Rejection Sampling Schemes for the two material configuration, $p_1/p_2 = 0.75/0.25$ 50
8.5 GPOKS Variation performance 51
8.6 Variance in the performance of the GPOKS Variations for the two material configuration. 51
8.7 White noise sensitivity in GPOKS Variations for the two material configuration. 52
8.8 Average #updates for the GPOKS Contenders in a two material setting. 53
8.9 Variance in the performance of the GPOKS Contenders in a two material configuration: $p_1/p_2 = 0.75/0.25$. 53
8.10 Average #updates using 8 web-pages. 54
8.11 Variance for the different algorithms using 8 web-pages with $\alpha/\beta = 0.6/1.0$ 55

1 Introduction

Parts of this thesis are published in [14], as a part of the thesis process, where it received the award for best student paper, and can be found in Appendix B **TODO: add the paper**.

1.1 Background and Motivation

Effective resource allocation is a topic of interest in most large scale systems, due to the simple fact that one *could* perform a set of tasks given a particular allocation of resources, one might save a lot of resources only by modifying the allocation scheme (i.e. how much resources is available to each task) and not the solution itself.

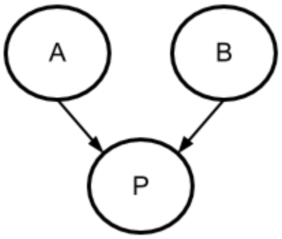


Figure 1.1: P depends on both A and B.

An example of this could be the allocation of running time for a distributed program P. Say that P needs the results from both task A and B in order to finish its calculations, how should it allocate running time to both of these (since P is a distributed program we may assume that A and B are running in parallel). A naive implementation might divide the running time uniformly among A and B, however this is only optimal in the case where A and B require an equal amount of running time to finish. For cases where this requirement does not hold, the running time should be allocated in such a manner that A and B both finish at the same time. Note that the gain in performance from this type of optimization does not depend on *how* A and B are solved, only the allocated resources have to do so, see figure 1.1 for a more visual representation of this example.

The challenge with these type of problems is there inherit stochastic nature. Continuing the above example, an exact solution would require us knowing in advance the time needed to finish processing both A and B as a function of their allocated running time. But for most non-trivial problems this type of information is not readily available and we must either estimate [1] it, or utilize some sort of adaptive scheme. A adaptive scheme would in this case start out

[1] Empirically or by exploiting known facts surrounding the problem.

with *a priori* estimate and gradually, based on observations, adapt towards a more optimal allocation.

1.2 Application: Web-polling

The Internet can be seen as a massive collection of ever changing information, continuously evolving as web resources are created, edited, deleted, and replaced [36]. In many cases obtaining sufficient information from the Internet is of uttermost importance, including social media monitoring, counter-terrorism and business intelligence. To keep up with these data requirements applied search engines have to constantly monitor available web resources to quickly and accurately detect changes, typically employing a polling regime.

The problem then becomes; how can we balance the available polling-resources in such a manner that we detect and retrieve the maximum amount of information? This problem was largely ignored in the literature until the introduction of the Learning Automata Knapsack Game (LAKG) [16] in 2006 where Granmo et al. also gave a formal problem definition, the so-called *Stochastic Non-linear Equality Fractional Knapsack* (NEFK) problem. In [19] the Hierarchy of Twofold Resource Allocation Automata (H-TRAA) algorithm was proposed, outstripping LAKG as the *state-of-the-art* for scenarios with a high number of unique resources.

Up to this point, the typical and also the simplest approach was to apply a uniform polling policy, where every web-resource was polled using a equal, fixed frequency. Clearly, this is a sub-optimal solution since each web resource may have a different characteristic. A web resource that rarely updates would be allocated a too high frequency and as such result in a large number of poll request returning without any new information. In addition it would deny a rapidly changing web resource the needed capacity to retrieve the available data. Thus, balancing the available polling capacity based on the web resources individual characteristic can result in a gain in information without increasing the total capacity of the system.

One can classify the type of solutions to this problem into two categories, online and estimation based solutions. The estimation based solution divide the allocation process into two phases. Firstly it will apply the uniform policy to all the web resources and monitor their update rates, gaining an estimate on there update probability. Threating these estimated values as the true underlying update probabilities for each web resource, one can use Lagrange Multipliers to calculate an estimated optimal resource allocation for the web resources.

However, while this algorithm can achieve an optimal balance, doing so would also require an arbitrary long estimation phase. That is, one either has to accept a sub-optimal solution due to an insufficient estimation phase, or wait for an extensive time period while the estimation process occurs to find more accurate update probabilities to base the calculations off.

In this thesis, we will be largely concerned with the on-line type of solution, here the algorithm perform both the estimation and calculation step at the same time, gradually improving itself.

This thesis introduces Gaussian Process based Optimistic Knapsack Sampling (GPOKS) – a novel online scheme for solving stochastic knapsack problems founded, as the name implies, on Gaussian Process (GP) [40] [5] based Thompson Sampling [50] enchanted by the principle of *Optimistic Thompson Sampling* [28].

1.3 Thesis definition

The Internet can be seen as a massive collection of ever-changing information, continuously evolving as web resources are created, edited, deleted, and replaced. Obtaining adequate amount of information is in many cases of paramount importance, typically task such as social media analysis depends not only on the quality of the data but also on the quantity and freshness of the data (e.g. the difference between the generation and the collection of the data). Most real-world scenarios boils down to being able to quickly and accurately detect when a web resource changes, as to only retrieve data that is new, reducing extraneous work for the collectors. Usually some sort of polling scheme is employed, where each web-page is polled for updates with a given frequency. The problem comes in the form of determining the optimal polling frequency for each web resource given the constraints of the total polling capacity of the system. The trivial and clearly sub-optimal solution would be to ignore the individual characteristics of each web resource and divide the total polling capacity uniformly among the web resources being monitored. While other methods have been suggested in the literature, the state-of-the-art polling scheme is LAKG, a Learning Automata based stochastic on-line algorithm for solving the Stochastic Nonlinear Equality Knapsack (SNEFK) whom the web-polling problem can be seen as a realization of. In this master thesis we will combine the principle of Optimistic Thompson Sampling (OTS) and Gaussian Processes (GP) to form a novel scheme for solving the SNEFK. We will then compare this new solution with existing state-of-the art techniques (LAKG) for solving SNEFK, with emphasis on the web-polling problem using a wide variety of configuration and scenarios.

1.4 Summary of Contributions

- We introduce the concept of Thompson Function Sampling applied to Gaussian Processes
- Connect the state-of-art solvers for Multi-Armed-Bandit problems to the Stochastic Nonlinear Equality Fractional Knapsack Problem
- Give a detailed performance centric comparison of variations of the GPOKS scheme.
- Compare the performance between GPOKS and the current state-of-the-art algorithms.

- Utilizing the restrictions GPOKS places on the GP we obtain a much faster calculation method for GPOKS than traditional GPs can offer.

1.5 Outline

The layout of this thesis can be summarized as follows:

- In Section 2 we introduce the formal problem description and the web-polling application.
- In Section 3 and 4 we cover the necessary background information.
- In Section 5 we give a brief overview of the state-of-art for solving the SNEFK problem.
- In Section 6 the novel GPOKS algorithm is introduced.
- In Section 7 we define the experiments that will be performed, and in Section 8 we report the results of the experiments.
- In Section 9 the conclusion and future work is covered.
- Appendix A gives a novel proof for the DET-KS algorithm.
- Appendix B is the published paper covering parts of this thesis.

2 Knapsack Problems

In this section, we formalize and introduce the Stochastic Nonlinear Equality Fractional Knapsack (SNEFK) Problem. First, by reviewing some more traditional variations of the knapsack problem, then we extend these into SNEFK that form the basis for GPOKS, we finish by showing how SNEFK can be used as a basis for the optimal web-polling problem.

2.1 Traditional Knapsack problems

The iconic knapsack problem is informally stated as: A man is going to the market, he has a single knapsack [2] to carry goods. Before he leaves he must decide what he should put into his knapsack to maximize his earnings at the market. Some items might be worth a lot, but will take up a lot of space, other less-valuable items may be worth less but will also take up less space. As such, a optimal solution take both of these views into account and finds the perfect setting that is just right.

[2] Knapsack: *A bag with shoulder straps, carried on the back, and typically made of canvas or other weatherproof material.*

{0-1} Knapsack Problem:
We now consider the classical {0-1} knapsack problem. This NP-hard problem [44] is studied extensively throughout the literature and several algorithms for finding the exact solution exists for *nice* problems [27] [37].[3] Here a subset of n items have to be packed into a knapsack with a capacity c. Each items has a value v_i and a weight w_i and the problem is then to select a subset of the items such that the total weight of the items does not exceed c and whose total value is a maximum. We can transform this into a Integer Linear Programing (ILP) model by first, without loss of generality assume that all v_i, w_i and c are positive integers. Introducing a binary variable s_i that takes on the value '1' if item i is in the knapsack and '0' otherwise, we arrive at the following mathematical definition:

$$\begin{array}{ll} \text{maximize} & \sum\limits_{i=0}^{n} s_i v_i \\ \text{subject to} & \sum\limits_{i=0}^{n} s_i w_i \leq c \end{array}$$

Unbounded Knapsack Problem:
The natural extension to this problem is the *Unbounded Knapsack Problem (UKP)*, here the restriction that we either select the item or not is relaxed into allowing s_i to take on any non-negative value i.e. we can select multiple copies of the same object as long as we respect the total capacity of the knapsack. This variation of the knapsack problem is also NP-hard [2] and have several applications such as network planning, network routing and parallel scheduling etc. [24]. Formally, UKP is expressed as:

$$\begin{array}{ll} \text{maximize} & \sum\limits_{i=0}^{n} s_i v_i \\ \text{subject to} & \sum\limits_{i=0}^{n} s_i w_i \leq c \;\; \forall i,\; s_i \in \mathbb{Z}^+ \cup \{0\} \end{array}$$

Linear Fractional Knapsack Problem:
Many real world allocation schemes operate in fractions, a large company might want to share its bandwidth smartly, a system might want to allocate CPU-time efficiently for different processes, or divide access to a shared buss. Again, the common denominator of these problem, is that they are operating using fractions instead of binary decisions.

The Linear Fractional Knapsack Problem (FKP) is a continuous knapsack problem as opposed to both 0-1 and UKP which are discrete integer problems. If we alter the knapsack to allow a real-valued amount of a material (item), denoted x_i, and translate the item value into a material value per unit we end up with UKP. We assume that both the weight w_i and material unit value v_i are pro-rated, and non-negative.

[3] Problems with exponentially growing coefficients cannot be solved efficiently, this is only natural considering the NP-hardness of the problem.

$$
\begin{array}{ll}
\text{maximize} & \sum_{i=0}^{n} x_i v_i \\
\text{subject to} & \sum_{i=0}^{n} x_i w_i \leq c \ \ \forall i,\ 0 \leq x_i \leq 1
\end{array}
$$

It is well know that FKP can be solved in a greedy manner, first we order the materials by their value density (v_i/w_i), and then we put the materials into the knapsack, one by one, based on their density, until the knapsack is full. If the last item cannot fit in its entirety we instead put a fraction of the material such that we fill the knapsack completely [6].

A common requirement for a fractional knapsack is that is must operate with limited set of information, and gradually improve the solution as one gain access to more information, an example of such an algorithm is found in [34].

2.2 The Nonlinear Equality FK (NEFK) Problem

One important extension of the classical FK problem is the Nonlinear Equality FK problem with a separable and concave objective function. The problem can be stated as follows [22]:

$$
\begin{array}{ll}
\text{maximize} & f(\mathbf{x}) = \sum_{i=0}^{n} f_i(x_i) \\
\text{subject to} & \sum_{i=0}^{n} x_i = c, \ \ \forall i, \in \{1, 2, \ldots, n\}\ , \ x_i \geq 0.
\end{array}
$$

Since the objective vector function $f(\mathbf{x})$ is considered to be concave, the value function $f_i(x_i)$ of each material is also concave. This means that the derivatives of the material value functions $f_i(x_i)$ with respect to x_i (hereafter denoted f_i') are non-increasing[4]. In other words, the material value per unit volume is no longer constant as in the linear case, but decreases with the material amount, and so the optimization problem becomes:

$$
\begin{array}{ll}
\text{maximize} & f(\mathbf{x}) = \sum_{i=0}^{n} f_i(x_i) \\
\text{subject to} & \sum_{i=0}^{n} x_i = c, \quad \forall i, \in \{1, 2, \ldots, n\}\ , \ x_i \geq 0.
\end{array}
$$

where $f_i(x_i) = \int_0^{x_i} f_i'(x_i)\mathrm{d}x_i$.

Efficient solutions to the latter problem, based on the principle of Lagrange multipliers, have been devised. In short, the optimal value occurs when the derivatives f' of the material value functions are equal, subject to the knapsack

[4] This can be seen by considering $f_i(x_i)$ as the *position* of a object, then if $f_i(x_i)$ is concave it must per definition have a negative double derivative (*acceleration*) $f_i''(x_i) < 0$ implying that the speed ($f_i'(x_i)$) is decreasing for each time step (x_i). Thus it is non-increasing.

constraints, identified as [8]:

$$\begin{array}{rl} & f_1'(x_1) = f_2'(x_2) = \ldots = f_n'(x_n) \\ \text{subject to} & \sum\limits_{i=0}^{n} x_i = c, \quad \forall i, \in \{1, 2, \ldots, n\} \ , \ x_i \geq 0 \end{array} \tag{2.1}$$

Example 2.1. Since this part is crucial for the later development of GPOKS in section 6 we present a concrete example of how equation 2.1 works. We use the following two concave materials functions:

$$\begin{array}{lll} f_1(x_1) = -x_1^2 & \Rightarrow f_1'(x_1) = -x_1^2 & \Rightarrow f_1''(x_1) = -2 \\ f_2(x_2) = -3x_2^2 & \Rightarrow f_2'(x_2) = -6x_2^2 & \Rightarrow f_2''(x_2) = -6 \\ \text{where} & x_1 \geq 0 \ \ , \ \ x_2 \geq 0 & \ \ \text{and} \ \ c = 1 \end{array}$$

Both $f_1(x_1)$ and $f_2(x_2)$ are clearly concave since their double derivative is negative. Since we only have two materials we can write: $x_2 = c - x_1 = 1 - x_1$. Attacking the problem directly employing Calculus we find $f(\mathbf{x}) = f_1(x_1) + f_2(x_2) = -x_1^2 - 3x_2^2 = -x_1^2 - 3(1 - x_1)^2 = -4x_1 + 6x_1 - 3$. We then calculate $f'(\mathbf{x}) = 6 - 8x_1$ and set f' equal to zero and solve for x_1 to give us: $x_1 = \frac{3}{4}$ and $x_2 = 1 - x_1 = \frac{1}{4}$.

Approaching the problem using equation 2.1 allow us to set $f_1'(x_1) = f_2'(x_2)$ again using the fact that $x_2 = 1 - x_1$ we find: $f_1'(x_1) = f_2'(x_2) \rightarrow -x_1^2 = -6x_2^2 \rightarrow -2x_1 = -6(1 - x_1) \rightarrow 6 = 8x_1 \rightarrow x_1 = \frac{3}{4}$ the same as when we used regular calculus.

See figure 2.1 for a plot of $f(\mathbf{x}) = -4x_1 + 6x_1 - 3$. Notice that if you want to move x_2 closer to '0' you also have to move x_1 closer to '1', and since their derivatives are equal, you would loose from doing so.

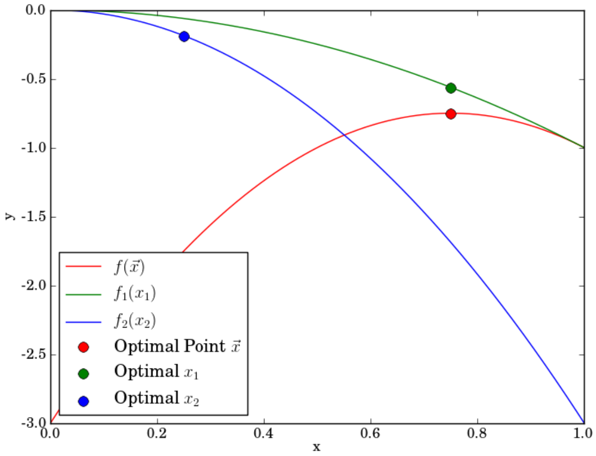


Figure 2.1: NEFK Example

2.3 The Stochastic Nonlinear Equality Knapsack (SNEFK) Problem

In this this thesis we generalize the nonlinear equality knapsack problem. First of all, we let the material value per unit volume for any x_i be a probability function $p_i(x_i)$. Furthermore, we consider the distribution of $p_i(x_i)$ to be a fixed unknown function. That is, each time an amount of x_i of material i is added to the knapsack, we only observe an instantiation of $p_i(x_i)$ and not the function $p_i(x_i)$ itself.[5] Given this stochastic environment, we then intend to devise an on-line incremental scheme that learns the mix of material that maximizes the expected value of the knapsack, through a series of informed guesses.

Thus, to clarify issues we are provided with a set of n materials that should be used to fill a knapsack with a fixed capacity c. However, unlike the NEFK in the Stochastic NEFK the material value for each material is a random quantity:

$$\text{material unit value}_i = \begin{cases} 1 & \text{with probability } p_i(x_i) \\ 0 & \text{with probability } 1 - p_i(x_i) \end{cases}.$$

As an additional complication, $p_i(x_i)$ is nonlinear in the sense that it decreases monotonically with x_i, i.e. $x_{i_1} \leq x_{i_2} \Longleftrightarrow p_i(x_{i_1}) \leq p_i(x_{i_2})$.

Since unit volume values are a random quantity, we operate with the expected unit volume values instead of the actual unit volume values. With this understanding, and the above perspective in mind, the expected of the amount x_i of material i, $1 \leq i \leq n$ becomes $f_i(x_i) = \int_0^{x_i} p_i(x_i) \mathrm{d}x_i$. Accordingly, the expected value per unit volume [6] of material i becomes $f_i'(x_i) = p_i(x_i)$. In this stochastic and non-linear version of the FK problem, the objective is to fill the knapsack such that the expected value $f(\mathbf{x}) = \sum_1^n f_i(x_i)$ of the materials contained in the knapsack is maximized. Thus, we aim to:

$$\begin{array}{ll} \text{maximize} & f(\mathbf{x}) = \sum\limits_{i=0}^{n} f_i(x_i) \\ & \text{where } f_i(x_i) = \int_0^{x_i} p_i(x_i)\mathrm{d}x_i, p_i(x_i) = f_i'(x_i) \\ \text{subject to} & \sum\limits_{i=0}^{n} x_i = c, \quad \forall i, \in \{1, 2, \ldots, n\}\ , \ x_i \geq 0 \end{array}$$

A fascinating property of the above problem is that the amount of information available to the decision maker is limited − the decision maker is only allowed to observe the current unit value of each material (either 0 or 1). That is, each time a material mix is added to the knapsack, the unit

[5] For the sake of consistency with previous work on the Stochastic NEFK Problem, we here model stochastic material unit values using Bernoulli trials. However, since GPOKS is based on Gaussian Processes, the central limit theorem opens up for addressing a number of other distributions too. Furthermore, there exist dedicated kernel functions for a variety of distributions.

[6] We hereafter use $f_i'(x_i)$ to denote the derivative of the expected value function $f_i(x_i)$ with respect to x_i.

value of each material is provided to the decision maker. The actual outcome probabilities $p_i(x_i)$, $1 \leq i \leq N$ however, remains *unknown* to the decision maker who have to relay on a sequence of material value amounts i.e. $(x_1, x_2, \ldots, x_n) \mapsto (v_1, v_2, \ldots, v_n)$ $\forall v_i, v_i \in \{0, 1\}$. As a direct consequence of this, the maximum material mix must be found through a series of trial-and-error, i.e. by experimenting with different material mixes and by observing the outcome and gradually work towards a maximum mixture.

2.4 Web-Polling as a Stochastic NEFK problem

As introduced in section 1.2 we will now consider the application of GPOKS towards the optimal web-polling problem. This web polling scenario is based on the setting found in [16] [19] and is based on the SNEFK problem.

Web resource monitoring primarily consist of repeatedly polling a set of web resources so that any change in the resources can be detected. In practical terms, there are several limiting factors that govern how such a scheme may operate e.g. processing power, bandwidth limitation. This impose a constraint on the *maximum* number of web resources that can be polled per time step. Since only a sub-set of the total resources can be polled at any given time step the problem turn into selecting this sub-set in a manner that maximize some metric. A natural metric in this case would be the number of updates detected.

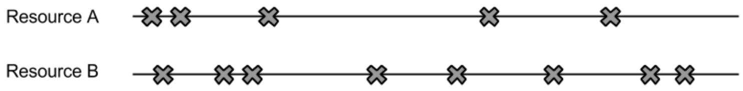


Figure 2.2: Time-line for two web-resources, a 'x' denotes a update for that resource.

Looking at figure 2.2 we see that *resource A* clearly [7] have a lower update probability than *resource B*. Moreover, the probability of detecting an update increases as a function of the delta time between each poll any particular resource. This is a crucial observation and as noted elsewhere [16] [17] [18] gives us the property that while the probability of detecting a change increase with the delta time between polls. The probability *decrease monotonically* with an increased polling frequency i.e. the inverse of the delta time. This is trivial to see when taken to the extreme, polling a web resource every time unit will yield a much lower probability to detect an update per poll than a web resource that is polled only at the very end of our time interval.

While other metrics for comparing the performance between web polling algorithms exists in the literature [36] [51] and we recognize that they all center around the *update detection probability* i.e. the chance of a update being discovered. Therefore, will we use the update detection probability as our primary

[7] In the given time interval or horizon.

token of interest. We introduce the notion of discrete time as a strictly increasing positive integer sequence, where the interval between the integers represents our atomic unit of decision making. In each time interval every web resource i has a constant probability p_i of being updated with new information. Furthermore, when a web resource is updated/changed any un-retrieved update is considered lost e.g. a news site where newer articles replace older ones on the front-page.

This behavior can be modeled as an exponential sequence, denote $q_i = 1 - p_i$ and define the frequency as x_i. Then $d_i(x_i)$ represent the update probability for web resource i using a polling frequency equal to x_i

$$d_i(x_i) \triangleq 1 - q_i^{1/x_i}. \tag{2.2}$$

By way of example, consider the scenario where a web resource remains unchanged in any single time step with probability 0.5. Then polling the web resource uncovers new information with probability $10.53 = 0.875$ if the web resource is polled every $3^{rd}+$ time step (i.e., with frequency $\frac{1}{3}$) and $10.5 = 0.75$ if the web resource is polled every 2^{nd} time step. As can be seen, increasing the polling frequency reduces the probability of discovering new information on each polling.

We also present a modified version (equation: 2.3) where we augment the $d_i(x_i)$ function with a white noise factor σ_{ws}, this version is used to see how algorithms respond to varying level of noise

$$d_i(x_i, \sigma_{ws}^2) \triangleq (1 - q_i^{1/x_i}) + \mathcal{N}(0, \sigma_{ws}^2). \tag{2.3}$$

Given the above considerations, our aim is to find the resource polling frequencies vector $\mathbf{x}$ that maximize the expected number of pollings uncovering new information per time step:

$$\begin{array}{ll} \text{maximize} & \sum\limits_{i=1}^{n} x_i d_i(x_i) \\ \text{subject to} & \sum\limits_{i=0}^{n} x_i = c \\ \text{where} & \forall i \in \mathbb{Z}^+, x_i \geq 0 \end{array}$$

3 Reinforcement Learning

This section will give a brief introduction to the field of Reinforcement Learning (RL) in a Multi Armed Bandit context. Then we will present two solution algorithms that form the basis for our stochastic knapsack scheme developed in section 2.

The essence of RL can be found in figure 3.1 and can be read as follows: An action a_t is selected at time t, we perform the action a_t on the environment, and the environment rewards us with some reward b_t. We then have to make a choice of what action we would like to perform next based on b_t, nevertheless we select a new action a_{t+1} and repeat the process. The interesting bit here is how exactly we select the new action to be performed. Usually the actions are selected in order to maximize the reward gained over some finite timespan.

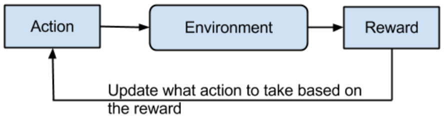


Figure 3.1: Reinforcement Learning

3.1 Learning Automata

In this section we will only briefly cover the topic of Learning Automata (LA) since a team of LA's form the basis for the *state-of-the-art* that we will be comparing our algorithm GPOKS against in section 8. An automaton is an entity very similar to the traditional Finite State Machine, and is defined in [33] as a quintuple $\{\underline{\Phi}, \underline{a}, \underline{\beta}, \mathcal{F}(\cdot,\cdot), \mathcal{G}(\cdot)\}$.

(i) *The state* of the automaton in instant t, denoted $\phi(t)$ is an element of the finite set

$$\underline{\Phi} = \{\phi_1, \phi_2, \dots, \phi_s\}. \tag{3.1}$$

(ii) *The action a* taken by the automaton in instant t, denoted $a(t)$ is an element of the finite set

$$\underline{a} = \{a_1, a_2, \dots, a_r\}. \tag{3.2}$$

(iii) *The input* of an automaton at the instant t, denoted $\beta(t)$ is an element of a set β, and is also the output or reward given by the environment. Here $\underline{\beta}$ can be defined both using a discrete set or a continuous interval on the real line, here $a, b \in \mathbb{R}$.

$$\underline{\beta} = \{\beta_1, \beta_2, \dots, \beta_m\} \text{ or } \underline{\beta} = (a, b). \tag{3.3}$$

(iv) *The transition function* $\mathcal{F}(\cdot,\cdot)$ determines the *next* state of the automaton at time instant $t+1$ and is thus a function of the current state $\phi(t)$ and the reward $\beta(t)$. We will present this function primarily in the form of transition graphs such as the one shown in Figure 3.2.

$$\phi(t+1) = \mathcal{F}[\phi(t), \beta(t)]. \tag{3.4}$$

Note that this definition of $\mathcal{F}$ allow it to be either deterministic or stochastic in nature.

(v) *The output function* $\mathcal{G}(\cdot)$ determines the *output* at state t in terms of the current state $\phi(t)$.

$$a(t+1) = \mathcal{G}[\phi(t)]. \tag{3.5}$$

Example 3.1. To finish up the section on LAs we will give a short example of a working automaton. This automaton will try to find what is best out of two actions thus $\underline{a} = \{a_1, a_2\}$ and we have a binary reward $\underline{\beta} = \{0, 1\}$. The transition function $\mathcal{F}$ for the case when $\beta(t) = 1$ and $\beta(t) = 0$ is given in figure 3.2. Note that we have defined a reward as $\beta = 1$ opposite of the regular LA literature notation as to fit notation wise with the rest of the thesis. The output function $\mathcal{G}$ is defined in equation 3.6, and the initial state is $\phi(0) = \phi_2$.

$$\mathcal{G}[\phi(t)] = \begin{cases} a_1, & \text{if } \phi(t) \in \{\phi_1, \phi_2\} \\ a_2, & \text{if } \phi(t) \in \{\phi_3, \phi_4\} \end{cases} \tag{3.6}$$

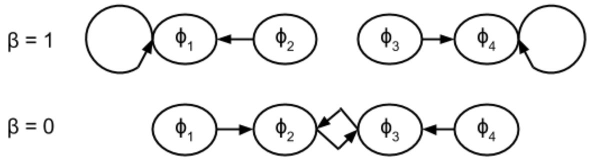


Figure 3.2: Transition function $\mathcal{F}[\phi(t), \beta(t)]$

Running the automaton

The idea behind this automaton is that for if we get a sequence of rewards we "remember" that this particular action performs well and as such we should be a bit lenient when it fails to offer a reward, but this effect should be proportional to the amount of previous rewards it has given us. In this specific case the memory for each action is two. The initial state of the automaton is defined as ϕ_2 and we find the current action as $a(t) = \mathcal{G}[\phi_2] = a_1$. We then perform the action a_1 on the environment, resulting in a reward $\beta(1) = 0$. The next state is then found by $\phi(2) = \mathcal{F}[\phi(1), \beta(1)] = \mathcal{F}[\phi_2, 0] = \phi_3$ and similarly the next action or output is $a(3) = \mathcal{G}[\phi(2)] = \mathcal{G}[\phi_3] = a_2$. Playing action a_2 yield a reward $\beta(2) = 1$, as such the updated state is $\phi(3) = \mathcal{F}[\phi(2), \beta(2)] = \mathcal{F}[\phi_3, 1] = \phi_4$.
Remark: The rewards given here is chosen to illustrate how a Learning Automata works, and should not be given any special consideration.

3.2 Thompson Sampling

The classical Thompson Sampling (TS) principle is closely tied to the exploration-exploitation dilemma found in the Multi-Arm Bandit Problem (MAB). As such we will first introduce the MAB problem then we will explain the Thompson Sampling in a MAB context. Later in Section 6 we will see how TS can be applied to a variant of the Knapsack Problem.

3.2.1 Multi Arm Bandit Problem

The Multi-Armed Bandit Problem (MABP) is perhaps the problem studied most in statistics [3], originally described by Robbins(1952) [41]. It captures the difficult trade-off between exploring and exploiting, that is, while exploring the environment for a good move, it should also empirically select the currently assumed best move. This problem turned out to be quite fundamental to the field of reinforcement learning [47] and genetic programing [20].

In this thesis, we will keep to a very basic formulation of the bandit problem. Given an environment consisting of n stochastic variables $X_1, X_2, \ldots, X_n$, our goal is the find out which of these will give the highest expected value. In other words we want to find a k that maximizes:

$$k = \max_{0 \leq i \leq n} \mathbb{E}[X_i] \tag{3.7}$$

This can be seen as a slot-machine with n arms[8] $\{a_1, a_2, \ldots, a_n\}$, where the reward from pulling the i'th arm is governed by the random variable X_i. The player then wants to maximize his winnings by only playing the best arm. Since the player does not have any *a priori* knowledge of the internal distribution of the arms he will have to explore, he does this by pulling the arms one at a time. After each pull the player is given a reward from the environment. How can the player then maximize his total winnings? See figure 3.3 for a illustration of an single time step in a MAB-Problem.

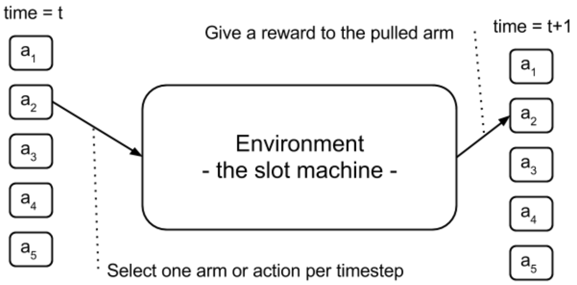


Figure 3.3: MABP: Action a_2 is selected in time t and given its reward, there are 5 available arms.

When first encountered with this problem, the first solution is generally some sort of greedy approach. In the literature, we find several greedy based algorithms, where the most notorious ones are, *greedy* and ϵ-*greedy*. The greedy algorithm calculates the average for each arm and continue to pull the arm with the highest average. Since greedy will only pull the arm with the highest average it has a tendency too explore to little and exploit too much. ϵ-greedy tries to alleviate this by introducing an exploration factor, namely with probability ϵ selected a random arm, and with probability $1-\epsilon$ select the arm with the highest average.

While these greedy algorithms work, there are better alternatives. The perhaps most notorious non-greedy MABP algorithm is UCB-1 [3] that builds upon

[8] Hence the name; Multi-Armed-Bandit

the idea of playing the arm with the highest upper confidence interval[9]. UCB-1 provides good empirical results but is also analytically tractable and tight finite time performance bounds are found in [3] and later improved on in [4].

3.2.2 Applications of MABP

There exists a wide variety of applications for the MABP, as such we will only mention a few noteworthy applications here. Rusmevichientong et. al [43] developed a scheme based on MAB that optimized the advertisement content a search-engine would display to its user based on the search query, such as to maximize the chance that the user clicked on the advertisement. Within the context of Computer Go[10] a revolution was sparked from the development of Multi-Armed-Bandit Tree Search (UCT) [23], using Monte-Carlo simulation gave rise to a novel set of techniques [49] eventually resulting in a 1st place in the Computer Go World Championship in 2008 [48].

3.3 Thompson Sampling in a Bandit Context

The Thompson Sampling Principle in MABP starts by formalizing our *a-priori* belief surrounding each arm, with a probability distribution. Of course, these distributions only represents our knowledge and not the underlying probability distributions, as these are unknown. An important note here is that this forms a bijective mapping between the set of arms and the set of distributions i.e. each arm is associated with one and only one distribution, and vice versa.

Thompson Sampling then works by sampling a point from each distribution associated with an arm. We treat this value as the true value of the arm and simply select the arm that offers the highest reward. We then update the probability distribution of the arm that we selected using Bayes Theorem with the reward from the pull. A common way to perform this Bayesian update in efficient manner is by selecting a prior from the prior conjugate family (popular choices include the Gaussian distribution and Beta distribution). As a consequence the variance of the distribution associated with the optimal arm will decrease and we see that the sampled value of the arm will asymptotically go towards the mean of the true underlying distribution. We therefore say that Thompson Sampling is *greedy-in-limit* [28] i.e. in the limit we only select the most profitable arm. Another property proved in [28] is that in-theory TS never stop exploring, due to the fact that the variance can never actually reach zero except in the limit. However, in practice we will have a number-underflow when the variance exceeds the precision of the storage format, thus reaching zero. Before we give an example on how TS works, we mention that until Agrawal et. al [1] theoretical finite performance bounds for Thompson Sampling where missing from the literature, following suit, [21] gave the first asymptotic optimal finite time proof for Bernoulli bandits in a TS context. These papers, among

[9] Hence the name Upper Confidence Bound (UCB)

[10] The field of playing Go, a chess like game, using a computer

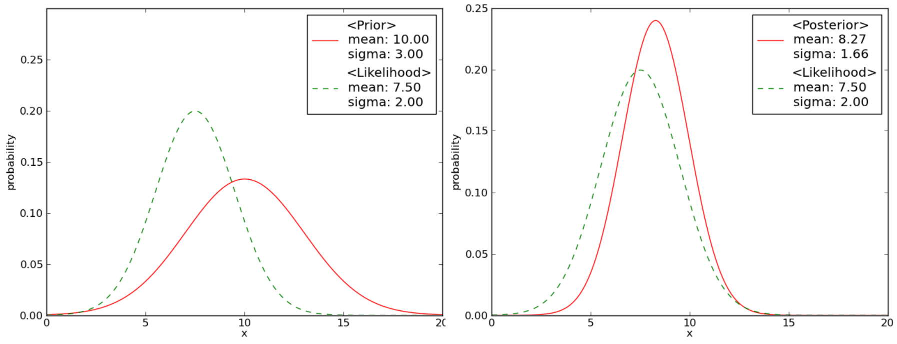
others, give a much sought-after theoretical foundation for the good empirical results of TS [15]. The algorithm for TS can be found later in section 3.3.1.

Example 3.2. To illustrate exactly how this conjugate prior update is performed we will illustrate a single step using a Gaussian prior, in figure 3.4 we start with a prior $\sim \mathcal{N}(\mu_{\text{pre}} = 10, \sigma_{\text{pre}} = 3.0)$ and receive a reward $m = 7.5$, where we have set a fixed likelihood standard deviation to $\sigma_{lh} = 2.0$ defining how much emphasis we should put on the new data i.e. the lower σ_{lh} is, the more we trust the data. The conjugate prior for a Gaussian distribution with a known likelihood variance is found in equation 3.8 (see [7] for a deviation).

$$\mu_{post} = \frac{(\sigma_{lh}^2\, \mu_{\text{pre}} + \sigma_{pre}^2 m)}{\sigma_{lh}^2 + \sigma_{pre}^2} \quad \text{and} \quad \sigma_{post}^2 = \frac{\sigma_{lh}^2 \sigma_{pre}^2}{(\sigma_{lh}^2 + \sigma_{pre}^2)} \tag{3.8}$$

Inserting our example values into equation 3.8 gives us a posterior distribution of $\mathcal{N}(\mu_{post} = 8.27, \sigma_{post} = 1.66)$.

Remark. Notice to the fact that the variance of the posterior is monotonically decreasing independent of the actual data in the likelihood. As such this form of TS is only directly applicable to stationary-MAB problems i.e., where the probability of an update follows some fixed unchanging distributions. One possible approach to overcome this limitation is described in [35] where a Sibling Kalman filter is applied to the hyper-parameters acting like a continuous sliding window.



Prior

Posterior

Figure 3.4: A Gaussian distribution prior and posterior

3.3.1 TS Algorithm

We here present the TS algorithm applied to a MAB problem using an unspecified conjugate prior. Let $\mathcal{U}(\theta,\ \mathbf{r})$ be a conjugate prior that takes a prior distribution (θ) and a likelihood $\mathbf{r}$ and returns a posterior distribution. Note that we without loss of generality augment θ to also contain any external parameters such as learning rate.

```
Algorithm: TS-MANB
Input: $\theta_0$ initial hyper-parameters; Number of arms $n$
Initialization: $d_0(\theta_0) = d_1(\theta_1) = \ldots d_n(\theta_n)$
Method:
For $t = 1, 2, \ldots$ Do
    1. For each Arm $j \in \{0, 1, \ldots, n\}$, draw a value $x_j$ randomly from the associated distribution $D_i$.
    2. Pull the Arm $i$ whose drawn value $x_i$ is the largest one:
           $$\alpha[t] = i = \underset{j \in \{0,1,\ldots,n\}}{\text{argmax}} \quad x_j.$$
    3. Receive a reward $\tilde{r}_i$ from pulling Arm $i$, and update hyper-parameters using the conjugate prior: $\theta_i = \mathcal{U}(\theta_i, \tilde{r}_i)$
End Algorithm: TS-MANB
```

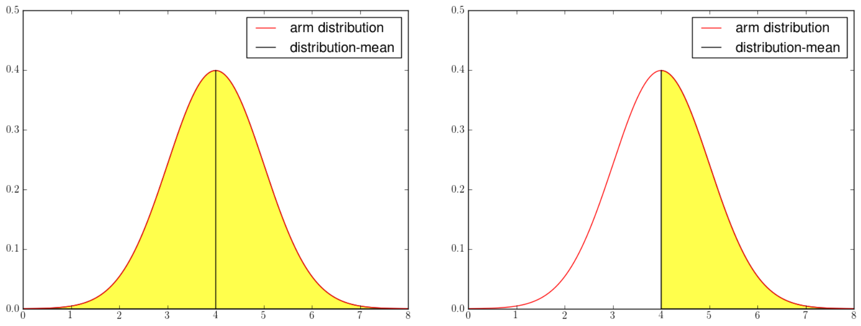
3.3.2 Optimistic Thompson Sampling

In both Thompson Sampling and UCB we find that an important incentive for exploration, is the practice of boosting the predictions of actions for which we are uncertain. However [28][11] takes this approach one step further. Recognizing that regular TS allow sampling of extreme values in both tails of the distribution, thus leading to a decrease in potential exploration. To alleviate this tendency *Optimistic* Thompson Sampling (OTS) was introduced, here the sample is clamped to the upper part of the distribution, see figure 3.5 for an illustration. This modification have show to compare favorable to UCB and TS in empirical studies [30] [9] for many practical configurations.

4 Gaussian Processes

This section start by introducing the notion of a function ensemble and it can be adapted to fit a set of data. The Bayesian Linear Model and feature space is then covered as a basis for the Gaussian Process. The rest of this chapter then covers various aspects of the usage of a Gaussian Process.

[11] First published as a technical report in [29].



Regular Thompson Sampling

Optimistic Thompson Sampling

Figure 3.5: TS and OTS, the potential sampling area is marked yellow.

4.1 Machine learning as a Function Ensemble

In most Machine Learning problems, the goal is to generalize from a finite set of observations (inductive learning), such that the uncertainty associated with making predictions decreases after taking observed data into account.

To this end, we postulate an *a priori* relationship between made observations and future observations. In practice this relationship is seldom without uncertainty and makes the task of generalization non-trivial. An example of a *non-trivial* relation could occur in time-series analysis where we want to predict future observations given a limited set of previous observations.

If we represent the postulated *priori* as an ensemble of functions, then the concept of learning transforms to one of adapting the ensemble to the observed data. One might imagine that the observations is *"generated"* by picking a function from the ensemble and that function is responsible for the observed data. We note that the observations them-self might be of a stochastic nature and that the ensemble might only hold a rough abstraction to the underlying model that we observe. This is, however, of little concern as long as we can obtain good predictions by conditioning the ensemble on the observations (probabilistic inversion) giving us a new *adapted* ensemble that is locked down on the observations while still variable everywhere else in the index set [45]. In the next section we will give a example on this effect.

We recognize and differentiate between two line of thoughts when it comes to choosing the ensemble: Using a *parametric model*, we restrict ourself to a single function class indexed by a finite fixed set of parameters. We induce an ensemble of functions by considering a distribution over the parameters in the index set. Learning then translate to adapting the parameters in such a manner that they fit the observed data. This methodology allow us to fully take advantage of an informed a priori postulate, for example, using expert knowledge. However, for cases where the function class does not match the observed data such an approach can lead to poor performance. In such cases a *non-parametric* model may give better performance.

A *non-parametric model* have the property that we do not have to worry about whether or not it is possible for the model to fit the data i.e. fitting a linear function to data originating from a sinusoidal function would yield poor results. Instead it is customary to specify some prior defining what functions are more probable than others. In the case of a Gaussian Process, we define a kernel or covariance function that specify how probable a function in our ensemble is, Section 4.4 covers this in greater detail.

Example 4.1. We finish this introduction with a simple example that shows how a adaption of the ensemble for the quadratic function $f(x) = 2x^3 - 3x + 1$ may be performed, note that we do not take any stochastic elements into account (making the probabilistic inversion step a boolean operator). For conceptual clarity, we will employ a *parametric* approach. That is, in Bayesian terms, we make use of a *prior* that discards any non-quadratic function from our ensemble[12] thus giving us a starting point of a ensemble containing only quadratic functions. In formal notations we may describe these steps as:

(1) Our ensemble set starts containing *all* possible one dimensional functions:
$E_1 = \{f | f : \mathcal{R} \to \mathcal{R}\}$

(2) We apply our prior, discarding any non-quadratic function.
$P_{\text{prior}} = \{f | f \text{ is a quadratic function}\}$
$E_2 = E_1 \cap P_{\text{prior}}$

Look at the first piece of observed data, $f^*(0) = 1$. The f^* here denotes the true unknown function that generated the data. We may then update our ensemble E_2 in such a manner that only quadratic functions that agree with the observations remain. Continuing from step (2):

(3) Our observation:
$O_1 = \{f^*(0) = 1\}$

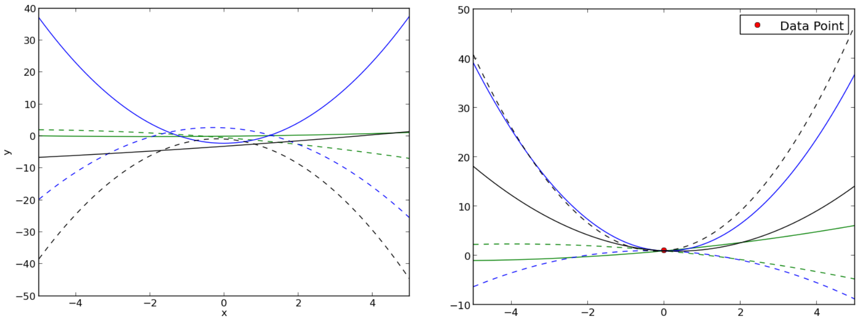
(4) Adapt our ensemble to fit with the observation
$E_3 = \{f | f \in E_2 \wedge f(0) = f^*(0)\}$

Visually we can see the difference between E_2 and E_3 in figure 4.1 where we have marked *some* elements from each set. Next we add two more observations, $f^*(2) = 3$ and $f^*(-1) = 6$ in step (5) and (6) respectively.

(5) Adapt our ensemble to fit with the observation
$E_4 = \{f | f \in E_3 \wedge f(2) = f^*(2)\}$

(6) Adapt our ensemble to fit with the observation
$E_5 = \{f | f \in E_4 \wedge f(-1) = f^*(-1)\}$

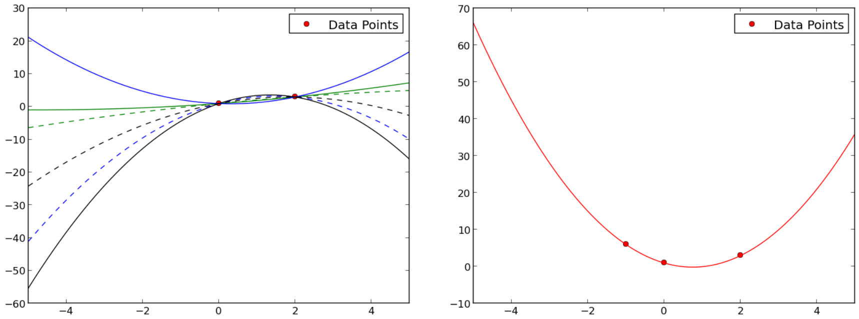
[12] Effectively we assume a-priori that the probability of a non-quadratic function generated the observed data is zero.



(A) Functions from E_2

(B) Functions from E_3

Figure 4.1: Ensemble adapting example step 2 and 4



(A) Functions from E_4

(B) The one function in E_5

Figure 4.2: Ensemble adapting example step 5 and 6

And we can see the how the ensemble adapts to the data in figure 4.2. Observe that $|E_5| = 1$ since we utilize a parametric-model and there is exactly one function that fit our data. However, if we picked a linear function to represent the data our model would not give us any exact solutions, and we would be forced to use some sort of probabilistic fitting such as least-squares regression to approximate an answer. A Gaussian Process alleviate these type of problems in a seamless manner due to its inherit non-parametric model [40] and thus forms the basis for a powerful machine learning tool.

4.2 Bayesian Linear Regression Model

At the heart of a Gaussian Process is the Bayesian take on the standard linear regression model with Gaussian white noise.

$$f(\mathbf{x}) = \mathbf{x}'\mathbf{w}, \quad y = f(\mathbf{x}) + \epsilon$$

Here $\mathbf{x}$ is the input vector and w is a vector of weights, notice that we have encoded the traditional bias term μ ($f(\mathbf{x}) = \mu + \mathbf{x}'\mathbf{w}$) implicit into the weight vector $\mathbf{w}$ by augmenting $\mathbf{x}$ with a additional element that is always one, thus the corresponding weight in w represents the customary bias term μ. As such we will ignore the bias term and instead work directly with the weight vector. Note that ϵ represents the white noise and is assumed to be independent identical distributed as a Gaussian distribution with zero mean and variance σ^2,

$$\epsilon \sim \mathcal{N}(0, \sigma^2).$$

This model therefore explicitly assumes that the difference between a observed value y and $f(\mathbf{x})$ can be explained by white noise (ϵ) and that the data is inherently linear. The resulting problem is then given a prior and a training set $\mathcal{D}$ to determine the *posteriori* estimate of $\mathbf{w}$.

Defining an observation training set with n entries as $\mathcal{D} = \{(\mathbf{x}_i, y_i)|i = 1, 2, \ldots, n\}$ where d is the dimension of $\mathbf{x_i}$, we can find a $n \times d$ *design matrix* X, similarly we define $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$. Using X and $\mathbf{w}$ we can now form the likelihood of observing a vector $\mathbf{y}$ as a product over each observation:

$$\begin{aligned} P(\mathbf{y}|X, \mathbf{w}) &= \prod_{i=1}^{n} P(y_i|\mathbf{x}_i, \mathbf{w}) \\ &= \prod_{i=1}^{n} P(y_i|N(\mathbf{x_i}'\mathbf{w}, \sigma^2)) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mathbf{x_i}'\mathbf{w})^2}{2\sigma^2}\right) \\ &= \frac{1}{(2\pi\sigma^2)^{n/2}} \exp\left(-\frac{1}{2\sigma^2}|\mathbf{y} - X'\mathbf{w}|^2\right) \\ &= \mathcal{N}(X'\mathbf{w}, \sigma^2 I) \end{aligned} \tag{4.1}$$

where I denotes an identity matrix of appropriate size and we observe that the likelihood is a multivariate Gaussian Distribution. Since we operate in a Bayesian manner we need to formulate a *prior* over the distribution of our parameters. This prior will encompass all our *belief* surrounding the parameters before considering the observations. Since we have a likelihood of the shape of a Gaussian distribution it is convenient to use a zero mean Gaussian prior with a covariance Σ_p over the weights,

$$\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \Sigma_p).$$

Then using Bayes theorem we can find the posterior distribution over the weights as:

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}$$

We have already found both the prior and the likelihood as such we need to find the marginal likelihood $P(\mathbf{y}|X)$. Since the marginal likelihood is independent of the weights, we see that it only represents a normalization constant and therefore we absorb it into the proportional part. After some algebraic manipulation [13] (the derivation can be found in [11]) gives us:

$$P(\mathbf{w}|X,\mathbf{y}) \sim \mathcal{N}(\bar{\mathbf{w}} = \sigma^{-2}A^{-1}X\mathbf{y}, A^{-1}) \tag{4.2}$$

where $A = \sigma^{-2}XX' + \Sigma_p^{-1}$.

Generating a predictive model (e.g. a way to predict future values) for the Bayesian Linear Regression at a test point $\mathbf{x}_*$ we utilize typical Bayesian methodology and simply average over all possible parameter values, weighted by their posterior probability, here $f_* \triangleq f(\mathbf{x}_*)$.

$$\begin{aligned} P(f_*|\mathbf{x}_*, X, \mathbf{y}) &= \int P(f_*|\mathbf{x}_*\mathbf{w})\ P(\mathbf{w}|X,\mathbf{y})\,\mathrm{d}\mathbf{w} \\ &= \mathcal{N}(\sigma^{-2}\mathbf{x}_*{}'A^{-1}X\mathbf{y},\ \mathbf{x}_*{}'A^{-1}\mathbf{x}_*) \end{aligned} \tag{4.3}$$

And the result is again a Gaussian, with a mean constructed from multiplying the posterior weight found in equation 4.2 with the test point.

Example 4.2. The function we would like to model is the Exponential Growth Model $g(t) = e^{\alpha+(\beta\times t)}$ found in for example Moore's Law [14]. Taking the logarithm on each, side gives us the linear equation $f(t) = \log g(t) = \alpha + (\beta \times t)$. We set $\alpha = 0.9$ and $\beta = 1.2$ as to give us practical values and generate some *"noisy"* samples ($\sigma = 0.2$), giving us a $\mathcal{D}$ found in the table below:

t:	0.2	0.8	1.4
y:	1.40	1.94	2.44

The data gives us a design matrix X (note that we have augmented the bias term into the $\mathbf{x_i}$ vectors):

$$\begin{bmatrix} 1.0 & 0.2 \\ 1.0 & 0.8 \\ 1.0 & 1.4 \end{bmatrix}$$

and a $\mathbf{y}$ vector equal to: $y = [1.40, 1.94, 2.44]'$. Applying equation 4.2 to this data gives us a Multivariate Gaussian Distribution as: $\mathcal{N}(\bar{\mathbf{w}}, A^{-1})$ where $\bar{\mathbf{w}} = [1.23, 0.86]$ and $A^{-1} =$

$$\begin{bmatrix} 0.048 & -0.043 \\ -0.043 & 0.055 \end{bmatrix}$$

To find our original expression, we substitute $\mathbf{w}$ back into our function $f(t)$ giving: $f(t) = 0.86 + 1.23t$. Transform $f(t)$ into $g(t)$ is done by applying the exponential function yielding: $g(t) = e^{f(t)} = e^{0.86+1.23t}$.

Making use of the predictive model (equation: 4.3), we want to find the distribution over the point $\mathbf{x}_* = [1.0, 0.5]$ (note that we again have to augment

[13] Basically writing out the parts that depends on the weight, "complete the square" and simplify until a Gaussian Distribution shape is achieved.

[14] Named after Intel co-founder Gordon E. Moore who in [32] noticed that the number of semiconductors doubled every 18 month.

the bias term). Gives a Gaussian distribution surrounding the point $\mathbf{x}_*$ where $f(\mathbf{x}_*) = f_* \sim \mathcal{N}(1.40, 0.018)$. Repeating this predictive process we can generate a plot over the predictive model as illustrated in Figure 4.3.

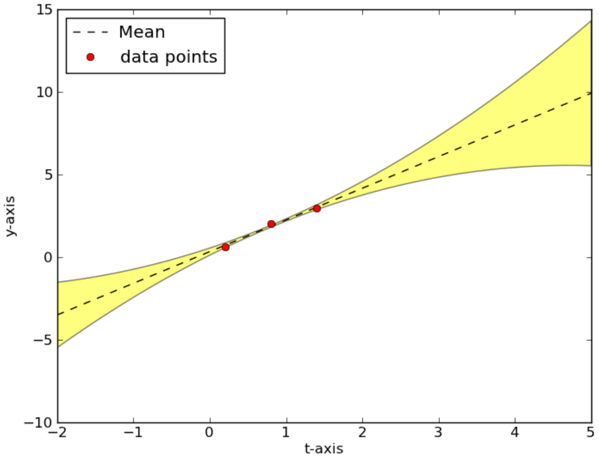


Figure 4.3: Bayesian Bivariate Regression, the yellow area marks a 95% confidence interval surrounding the mean of the predictive model.

4.3 Kernel-Induced Feature Spaces

Example 4.2 in the preceding section contains an important piece of information besides the multivariate Bayesian linear regression, in-fact, it introduces a light version of a powerful mathematical tool called *Kernel-Induced Feature Spaces*. In [31], Minsky and Papert highlighted the limitations of linear learning algorithms, such as the classical Rosenblatt's Perceptron algorithm [42]. (However, by lifting a non-linear problem into a feature space such that the problem is again linear, albeit in a different space, allows us to take full advantage of the power of linear algorithms.) A commonly employed preprocessing strategy in machine learning is changing the representation of the data, note that we transform from a d-dimensional input space to a r-dimensional feature space:

$$\mathbf{x} = [x_1, x_2, \ldots, x_d] \mapsto \Phi(\mathbf{x}) = [\Phi_1(\mathbf{x}), \Phi_2(\mathbf{x}), \ldots, \Phi_r(\mathbf{x})].$$

Again, referring back to example 4.2 we see that the mapping $\Phi(\cdot)$ used is the logarithmic function. If we now reformulate our multivariate Bayesian linear regression model in terms of this mapping $\phi(\cdot)$ we get:

$$f(\mathbf{x}) = \phi(\mathbf{x})'\mathbf{w}.$$

Here $\mathbf{w}$, the parameter vector, is extended to contain n parameters. Repeating the steps in the previous section only replacing all instances of X with $\Phi(X)$

(the *transformed*[15] design matrix) we obtain the following predictive model:

$$P(f_*|\mathbf{x}_*, X, \mathbf{y}) \sim \mathcal{N}(\sigma^{-2}\phi(\mathbf{x}_*)' A^{-1}\Phi\mathbf{y},\ \phi(\mathbf{x}_*)' A^{-1}\mathbf{x}_*)$$

where $\Phi = \Phi(X)$ and $A = \sigma^{-2}\Phi\Phi' + \Sigma_p^{-1}$.

Seeing that the above equation is defined solely in terms of the inner products in input space, then we can avoid any explicit computation of $\Phi\Phi'$ (the Gram Matrix) and instead define a kernel $K(\mathbf{x}, \mathbf{x}')$ that replaces the occurrences of the inner product space. This is commonly referred to as the *kernel trick* and form the basis of what we refer to as *kernel based learning methods* and is used in machine learning methods such as Support Vector Machines [10] and Gaussian Processes [45] [40].

4.4 Gaussian Process

The main point of a Gaussian Process (GP) is to adapt an ensemble of functions in a stochastic manner to fit a set of data points. We will now give a more formal introduction to how a GP works.

Definition 4.1. A Gaussian Process *is a collection of random variables, any finite number of which have a joint Gaussian distribution* [40] □

Let $f(\mathbf{x})$ denote the real-process that we model, then we define its mean function $m(\mathbf{x})$ and the kernel function $K(\mathbf{x}, \mathbf{x}')$ as:

$$\begin{array}{ll} m(\mathbf{x}) & = \mathbb{E}[f(\mathbf{x})] \\ K(\mathbf{x}, \mathbf{x}') & = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x})(f(\mathbf{x}') - m(\mathbf{x}')], \end{array}$$

and will write the Gaussian Process as

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), K(\mathbf{x}, \mathbf{x}')).$$

Gaussian process (GP) regression is a Bayesian approach which assumes a GP prior[16] over functions, i.e., assumes a priori that function values behave according to [39]:

$$P(\mathbf{f}|\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = \mathcal{N}(\mathbf{0}, K),$$

where $\mathbf{f} = [f_1, f_2, \ldots, f_n]'$ is a vector of latent functions values i.e. $f_i = f(\mathbf{x}_i)$. And K is a covariance matrix given by the kernel function, $K_{ij} = K(\mathbf{x}_i, \mathbf{x}_j)$. From this model, we realize that a Gaussian Process follows the Bayesian Multivariate Regression Model, where the number of dimensions is n, one per latent function value. Here each latent function value f_i at $\mathbf{x}_i$ is treated as a random variable which covary with an arbitrary point $\mathbf{x}_j$ as determined by the kernel function $K(\mathbf{x}_i, \mathbf{x}_j)$.

[15] Each vector x_i in the training set is separately lifted into the feature space and aggregated into a design matrix.

[16] For notational simplicity will we in this thesis exclusively use zero-mean priors ($m(\mathbf{x}) = \mathbf{0}$).

4.4.1 Squared Exponential Kernel

In this thesis we will only consider the one dimensional squared exponential kernel (SE Kernel) defined as:

$$K(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}(x_p - x_q)^2\right) + \sigma_n^2 \delta_{pq} \tag{4.4}$$

Here l is the characteristic length-scale parameter that determines how rapidly the correlation should decay as the distance between x_p and x_q increases, σ_f^2 is the signal variance and σ_n^2 is white noise (note that δ_{pq} here denotes the Kronecker delta between x_p and x_q). This kernel function is infinitely differentiable, giving it mean-square derivatives of all orders and is therefore *very* smooth [40], and it is perhaps the most commonly used kernel in machine learning literature. It can be shown [26] that it corresponds to a linear Bayesian regression model with an infinite number of basis functions, hence the need to work implicit in a feature space using the *kernel trick*.

For the hyper-parameters $l^2 = 1.0,\ \sigma_f^2 = 1.0,\ \sigma_n^2 = 0.1$. We present table 4.4 to provide some insights in how much different points covary, the x_p remains fixed at $x_p = 1.0$ and $K(x_q, x_q)$ is presented for different values of x_q.

	1.0	1.1	1.2	1.5	2.0	4.0
x_q	1.1	0.99	0.98	0.88	0.60	0.01

Figure 4.4: SE-Kernel covariance between fixed point $x_p = 1.0$ and x_q.

From this table we notice two things, firstly that if two points are placed in the same index (i.e. $x_p = x_q$) then the white noise component (σ_n^2) of the SE Kernel comes into play, this will form the basis for the efficient GP implementation of GPOKS in section 6.2. And secondly, the fact that the covariance values decrease rapidly with only a difference of three between x_p and x_q the observations does almost not affect each other.

4.4.2 GP Inference and Function Sampling

Inference in a GP model is in many ways equivalent to using the predictive model introduced for MultiVariate Gaussian Regression (MVGR). As with MVGR we start by defining a training set $\mathcal{D} = \{(\mathbf{x}_i, y_i),\ i = 0, 1, \ldots, n\}$, with n pairs of observed data. Here $\mathbf{x}_i$ is a vectorial point and f_i is the scalar output at $\mathbf{x}_i$, that is: $f_i = f(\mathbf{x}_i)$, to take noise[17] into account we formulate the following relationship:

$$y_i = f(\mathbf{x}_i) + \epsilon_i,\ \text{where } \epsilon_i \sim \mathcal{N}(0, \sigma_{noise}^2])$$

To predict the function value $\mathbf{f}_*$ at the points $\mathbf{x}_*$, we, use Bayes rule, to find the joint posterior $P(\mathbf{f}, \mathbf{f}_*|\mathbf{y})$ by combining the GP prior with the likelihood $P(\mathbf{y}|\mathbf{f}, \mathbf{f}_*)$ giving:

[17] As with MVGR we assume white i.i.d Gaussian noise

$$P(\mathbf{f}, \mathbf{f}_* | \mathbf{y}) = \frac{P(\mathbf{f}, \mathbf{f}_*) \times P(\mathbf{y}|\mathbf{f}, \mathbf{f}_*)}{P(\mathbf{y})}.$$

Marginalizing out the training data $\mathbf{f}$ then gives us:

$$P(\mathbf{f}_*|\mathbf{y}) = \int P(\mathbf{f}, \mathbf{f}_*|\mathbf{y}) \mathrm{d}\mathbf{f} = \frac{1}{P(\mathbf{y})} \int P(\mathbf{y}|\mathbf{f}) P(\mathbf{f},\ \mathbf{f}_*) \mathrm{d}\mathbf{f},$$

where the joint GP prior and likelihood can be written as:

$$P(\mathbf{f},\ \mathbf{f}_*) = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K_{\mathbf{f},\mathbf{f}} & K_{*,\mathbf{f}} \\ K_{\mathbf{f},*} & K_{*,*} \end{bmatrix}\right) \text{ and } P(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma^2_{noise} I),$$

where $K_{a,b}$ is defined as the covariance between a and b, and we use asterisk $*$ as a shorthand for $\mathbf{f}_*$. Since the joint GP prior and likelihood is both Gaussian we can evaluate the integral in a closed form giving us a Multivariate Gaussian predictive model as:

$$P(\mathbf{f}_*|\mathbf{y}) = \mathcal{N}(K_{*,\mathbf{f}}\ [K_{\mathbf{f},\mathbf{f}} + \sigma^2_{noise} I]^{-1}\mathbf{y},\ K_{*,*} - K_{*,\mathbf{f}}\ [K_{\mathbf{f},\mathbf{f}} + \sigma^2_{noise} I]^{-1}\ K_{\mathbf{f},*}) \tag{4.5}$$

Example 4.3. The following example will demonstrate how a GP operates: We start out with a non-informative prior distribution, illustrated in Figure 4.5. As seen in the figure, we can using this prior draw random function samples, note the mean function: $m(\mathbf{x}) = \mathbf{0}$. Next in Figure 4.6 we add a data point (or observation). The confidence interval surrounding the data point immediately shrinks, making the likelihood of a random sample functions passing close to the data point much higher.

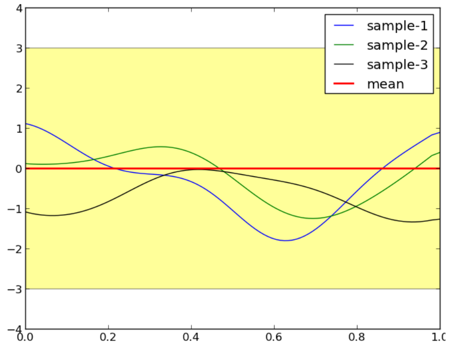


Figure 4.5: GP Prior distribution, the marked yellow area is a 95% confidence interval.

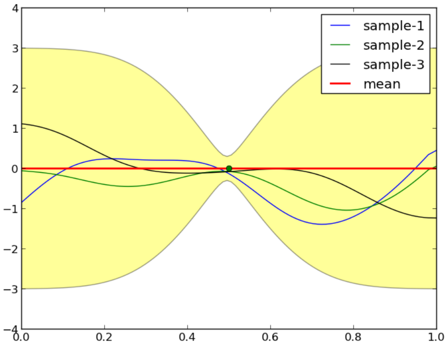


Figure 4.6: GP Posterior after 1 observation, the marked yellow area is a 95% confidence interval.

Now, we continue and add some additional data points, illustrated in Figure 4.7 and Figure 4.8. We can now observe *"bubbles"* between the data-points, where the confidence interval is higher. This indicates that the predictive model is more uncertain about the underlying function $f(\mathbf{x})$ in this area. We will later see how GP-UCB and several GPOKS Variations use this to explore the function space, looking for the highest *potential* point. It also interesting to see how similar this model looks to the Bayesian Linear Regression example given earlier, except that the GP handles non-linear function, due to its non-parametric nature.

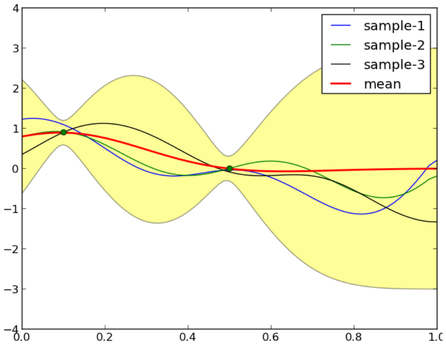


Figure 4.7: GP Posterior after 2 observations, the marked yellow area is a 95% confidence interval.

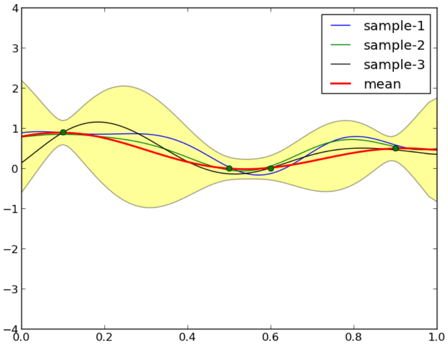


Figure 4.8: GP Posterior after 4 observation, the marked yellow area is a 95% confidence interval.

5 SFNEKS Problem - State-of-the-art solvers

The Learning Automata Knapsack Game (LAKG) algorithm first proposed in [16] is considered the state-of-the art for SFNEKS problems with relatively small scale configurations i.e., a low amount of different materials. However in [19] H-TRAA was introduced, with a significantly better convergence rate for large scale problems i.e., a hight amount of different materials. In this thesis, we will compare our algorithm GPOKS with both of these and will therefore give a condensed overview of their mechanics to better outline the differences.

5.1 LAKG – Learning Automata Knapsack Game

Learning Automata Knapsack Game (LAKG) was long the leading contender for solving the SFNEKS problem [16]. Here each material in a SFNEKS is represented with a separate LA, with N states $\underline{\Phi} = \{\phi_1, \ldots, \phi_n\}$. Loosely stated the amount of a single material is then the current state $\phi(t)/N$.

The *game* part of the LAKG comes from the fact that the LAs cooperate; All the LAs share a common variable, *is the knapsack full or not?*(i.e. Is the sum of all the LA states equal to 1?), This behavior is reflected in their transition function $\mathcal{F}$ for material i:

$$\Phi_i(t+1) = \Phi_i(t) + 1 \textbf{ Knapsack not full } \textbf{ and } \beta_i(t) = 1,$$

$$\Phi_i(t+1) = \Phi_i(t) - 1 \textbf{ Knapsack not empty and } \beta_i(t) = 0.$$

Since the initial state of this scheme is $\lfloor N/2 \rfloor$ the LAKG have an easier time coping with problems whose solution is close to uniform.

5.2 H-TRAA – Hierarchy of Twofold Resource Allocation Automata

Hierarchy of Twofold Resource Allocation Automata (H-TRAA) introduced in [19] outperforms LAKG in several areas, and is currently the *state-of-art* for large scale SFNEKS problems.

It starts by dividing the materials into pairs, where each pair is associated with a single learning automata. This LA controls the relative normalized weight between its two materials i.e. It may weight one 80% and one 20%. This can be seen as separate knapsack problem involving two materials. Now we pair the LAs together and again assign an LA to control the relative normalized weight between these two. We repeat this process to effectively form a binary tree where the leafs are the original LAs, each representing two materials. Note that this require us to have exactly $2^n, n \in \mathbb{Z}^+$ materials.

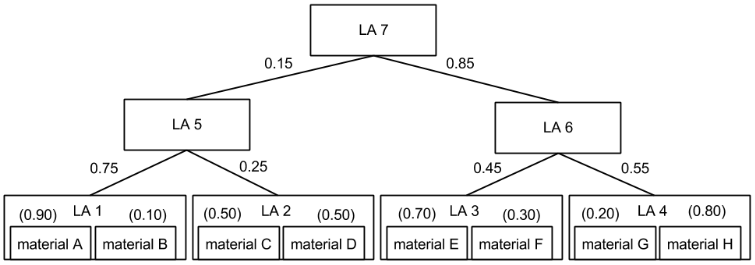


Figure 5.1: H-TRAA with 8 materials (A - H) and some example weights.

To find the amount of a material we start at the LA containing the material in question and work our way upwards to the root. Taking the product of the weights as we move along. In figure 5.1 one can observe a H-TRAA in work with eight materials (named A-H), and the amount of material A that we would like to add would then be: $x_A = 0.90 \times 0.75 \times 0.15 = 0.10125$. Note that the H-TRAA initially start with a uniform distribution over the materials.

The update (or reinforcement) phase is done as follows: If we added material i to the knapsack, and received a reward, first update the directly associated LA, then that LA will again update its parent LA with a reward. This processes continues until the root node is reached. As a example, if we got a reward when adding x_A of material A. We would first give a reward to LA-1, then LA-1 would give a update to LA-5 and LA-5 to LA-7.

The remaining piece of H-TRAA is the TRAA part. TRAA is the scheme used in each LA node. It has N states and like the LAKG suggest a material amount(s) equal to n/N and $1-n/N$ for the two materials $(i = 1, 2)$ respectively (here $n = \Phi(t)$ the current state). The big difference arise in the transition function $\mathcal{F}$:

$$\Phi(t+1) = \Phi(t) + 1 \textbf{ If rand() } \leq \ (1 - n/N) \ \textbf{ and } \ \beta_1(t) = 1 \ \textbf{ and } \ 1 \leq n \leq N$$

$$\Phi(t+1) = \Phi(t) - 1 \textbf{ If rand() } \leq \ (n/N) \ \textbf{ and } \ \beta_2(t) = 1 \ \textbf{ and } \ 1 \leq n \leq N$$

Since the chance of moving in one direction decreases the more one goes in that particular direction the TRAA will quickly take the first steps but will be slower to converge totally. This effect is enhanced due to the fact that if one child fails to propagate (i.e. its rand() value was too low) then its ancestors LAs will not get a chance to update its state.

5.3 GP-UCB – Gaussian Process Upper Confidence Bounds

This is a Bayesian Optimization (BO) scheme called GP-UCB that was introduced in [46] alongside with a strong theoretical foundation for its inner workings. The gist of the GP-UCB algorithm is that, unlike most MAB solvers, who assume that each arm is *independent* of the other arms, GP-UCB assume that there is a relationship between the different arms. By modeling the inter-arm relationship as a GP, where arm number i is represented in the GP as point x_i. Then by using a problem specific kernel, GP-UCB allow for a significant faster exploration as each observation obtained does not only affect a single arm, but instead affects all arms as specified by the kernel function.

To find the next arm to select GP-UCB perform Gaussian Regression over all the arms, and pick the point with the highest 95% confidence interval i.e. mean $+ 2\sigma$.

If this inter-arm relationship does not exist then GP-UCB is not suitable for the problem at hand and a regular UCB-1 could be tried instead. However, in the Stochastic NEFK problem we do have a smooth response surface, thus fulfilling the criteria needed to use GP-UCB.

5.3.1 Applying a MAB solver to the Stochastic NEFK

The MAB solver introduced in 3.2.1 as well as GP-UCB are not made to handle the SFNEKS problem directly, but if we translate the material amounts from a continuous value into a set of discrete values we can map each possible value to a arm, thus creating a MAB problem. The downside with this approach is that the number of arms will increase drastic as the number of materials increase. Say we have n materials and use k different discrete amounts (e.g. $k = 101 \Rightarrow \frac{1}{101-1} = 0.01$ intervals in x_i, the material amount)[18]. We can translate the problem into a simple combinatorial exercise:
How many solutions exists for the follow integer problem?

$$y_1 + y_2 + \ldots + y_n = k$$
$$\text{where } \ \forall i \in \mathbb{Z}^+, 1 \leq x_i \leq k \text{ and } x_i \in \mathbb{Z}$$

[18] The minus one part of $\frac{1}{101-1}$ comes from the fact that we need to include the endpoint as well as the starting point

Where the solution is simply: $\binom{k+n-n-1}{k-n} = \binom{k-1}{k-n}$. Note that in this formulation we disallow a zero material amount for any material, thus decreasing the number of available arms. As we see in table 5.2 the amount of arms required even for a small amount of materials discourage the application of MAB solver for problems using 3 materials or more. We will therefore only present MAB results for the cases with two materials.

k	*interval*	2 materials	3 materials	4 materials	5 materials
11	0.1	10	45	120	210
101	0.01	100	4950	161700	3921225

Figure 5.2: Number of arms when using MAB to solve SFNEKS

6 Gaussian Process based Optimistic Knapsack Sampling (GPOKS)

In the MAB Problem outlined in section 3.2.1, we pursue to find the best arm using a sequence of educated guesses, a so-called trial-and-error process. The GPOKS attempts something similar, however instead of simply seeking the best material (bandit arm), we are searching for a *mixture* of materials, also referred to as a mixture strategy in Game Theory [12].

We here present a novel GP based model for stochastic NEFK problems, where a *collection* of GPs capture the individual material unit values, forming a *model* of the problem. The GPOKS algorithm builds upon a famous quote by G. Pólya: *"If there is a problem you can't solve, then there is an easier problem you can solve: find it."* [38].

Using the GP collection based model, Thompson Sampling is applied to sample a likely deterministic NEFK problem instances from the GPs. That, in turn can be solved based on Lagrange Multipliers, producing a potential solution to the SNEFK problem at hand. Using this solution as a guide, we interact with the knapsack and update the collection of GPs, gradually minimizing the difference between the true underlying stochastic NEFK problem and the sampled deterministic NEFK problem. Therefore the difference between the solutions are also minimized. A conceptual overview of GPOKS can be seen in Figure 6.1.

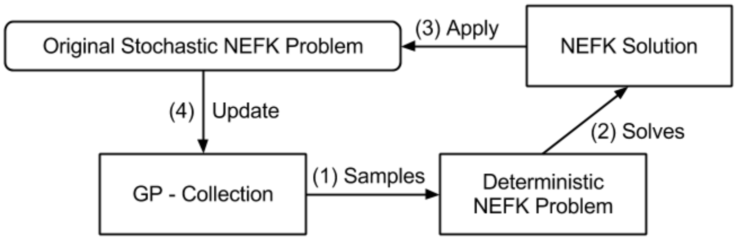


Figure 6.1: Conceptual overview of the different states of GPOKS; (1) Samples the GP based model for a similar deterministic NEFK problem. (2) Solves the NEFK problem. (3) Apply the solution to the NEFK problem to the original SNEFK problem. (4) Update the GP model to more closely represent the SNEFK problem. So that the next sample will be closer to the original problem.

6.1 Sampling and solving a NEFK problem

To represent the stochastic NEFK problem in a manner that allows us to sample a NEFK problem we associate one GP per material. Thus we gain a collection of GPs denoted $\mathcal{C} = \{GP_1, GP_2, \ldots, GP_n\}$ where n refers to the number of materials in the problem, where each $GP_i \in \mathcal{C}$ maps to a $f'(x_i) = p_i(x_i)$. For GP_i the x-axis will denote the material amount (x_i) and the y-axis will denote the material value probability function $p(x_i)$. Figure 6.7 shows one such GP.

Recalling the NEFK problem definition from Section 2.2, repeated here for clarity:

$$\begin{aligned} &\text{maximize} \quad && f(\mathbf{x}) = \sum_{i=0}^{n} f_i(x_i) \\ &\text{subject to} \quad && \sum_{i=0}^{n} x_i = c, \quad \forall i, \in \{1, 2, \ldots, n\}\ ,\ x_i \geq 0. \end{aligned}$$

We observe that the NEFK problem is completely defined using the set of $f_i(x_i)$ functions and its capacity. As such we need to extract these functions from the GP collection, one per material. In Section 4.4.2 we introduced the concept of function sampling from a GP. We use this technique to sample a function $\hat{p}_i(x_i)$ from GP_i and formulate a probable NEFK problem using these i.e. maximize $\sum \hat{p}_i(x_i)$ instead of $\sum p_i(x_i)$.

6.1.1 Solving NEFK

Having obtained a completely defined NEFK problem we need to solve it. To perform this task we propose the *Deterministic Knapsack Solver* (DET-KS) algorithm. DET-KS takes a set of concave material functions and iteratively calculate an optimal mixture vector M, the solution to the NEFK problem.

The way DET-KS works is that it starts with initial mixture vector set to some small value (as to disallow any solution not involving all materials). It then, in a greedy fashion, adds a small fraction of the knapsack capacity to the material that would gain the least from the increase in material amount, it repeats this step until the knapsack is full. We were unable to find any published proof of the convergence properties of DET-KS, we therefore present a novel proof in Appendix A.

```
Algorithm: DET-KS
Input: Set of functions $\mathbf{f} = \{f_1, \ldots, f_n\}$ and $c$ the knapsack capacity.
Initialization: $\mathbf{M}[1] = \ldots = \mathbf{M}[n] = \epsilon$;
# Typically $\epsilon$ can be set to some sufficiently small value.
Method:
While $sum(\mathbf{M}) =< c$ Do
    1. Find the material $i$ that have the smallest delta value weighted by its mixture:
           $i = \operatorname*{argmin}_{j \in \{1,n\}} [\mathbf{M}[j] \times f(\mathbf{M}[j])] - [(\mathbf{M}[j] + \epsilon) \times f(\mathbf{M}[j] + \epsilon)].$
    2. Increase the amount of $f_i$ in the mixture: $\mathbf{M}[i] = \mathbf{M}[i] + \epsilon$.
End While
Return M
End Algorithm: DET-KS
```

6.1.2 Rejection Sampling

To further enhance GPOKS we draw upon the success of Optimistic Thompson Sampling (OTS) (see section 3.3.2 for more info) and introduce the notion of *rejection sampling* for GPOKS. Rejection sampling or *acceptance-rejection method* is a method for sampling from a random distribution where the probability density function of the distribution makes direct sampling infeasible [25], and can typically be seen applied to (Bayesian) Monte Carlo Markov Chain posterior estimation techniques such as Gibbs Sampling [13].

In our case we are interested in sampling a function that fulfill some specified criteria (e.g. optimistic). We proceed by sampling a candidate function $\hat{f}$, if this fulfill the criteria then we are done, else we reject the candidate function and sample a new candidate function. We repeat this process as long as necessary to find a candidate that is acceptable, that is, fulfill the criteria. In GPOKS we will use two criteria for rejection sampling, *optimistic* and *monotonic* to refine the type of sample functions we use for our NEFK problem. For a baseline reference we include figure 6.2 where a set of functions are sampled from the GP *without* rejection sampling.

As long as the candidate sampling is done i.i.d. the chance of finding a specific sample function remains the same as if we had sampled from the true distribution.

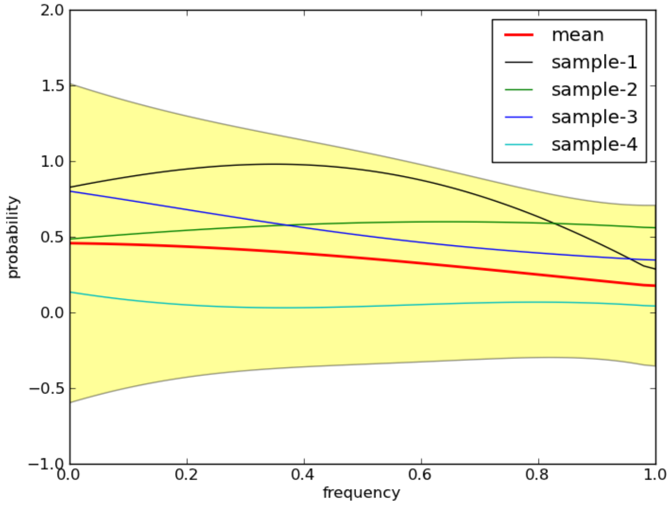


Figure 6.2: Samples from a GP without rejection sampling.

6.1.3 Optimistic Rejection Sampling

To generate *"optimistic"* sampling functions using rejection sampling we first have to define a criterion for what an optimistic function is. Given an index set

X and a mean function $\mu(x)$ we define an *optimistic* function $g(x)$ as a function that fulfill the criterion:

$$\forall x \in X, \;\; g(x) \geq \mu(x). \tag{6.1}$$

In more informal terms, it means that we are only interested in functions that strictly reside in the upper part of the confidence interval surrounding the mean function μ_i in GP_i. Figure 6.3 illustrates this point.

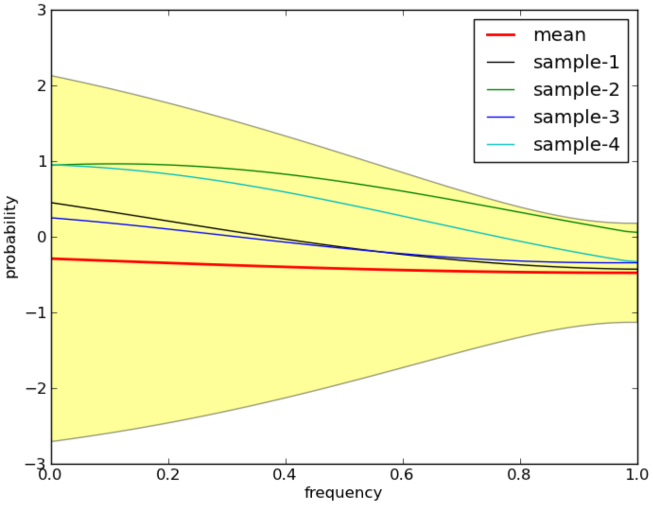


Figure 6.3: Samples from a GP using rejection sampling with a optimistic criterion.

6.1.4 Monotonic Rejection Sampling

In the definition of the NEFK and its stochastic variant SNEFK, each material volume function $p_i(x_i)$ is required to be non-increasing. Rejection sampling allows us to take full advantage of this fact by defining a criterion where this property is fulfilled. Given an index set X we define the criterion for a *monotonic* function $g(x)$ as:

$$\forall x_1, x_2 \in X, \; x_1 \leq x_2 \Rightarrow \;\; g(x_1) \geq g(x_2). \tag{6.2}$$

As seen in figure 6.4 the monotonic functions closely resembles the example NEFK material value function given in Section 6.1.

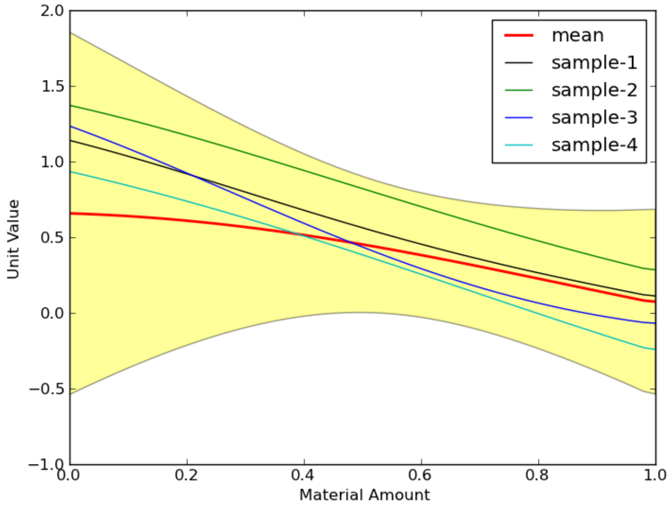


Figure 6.4: Samples from a GP using rejection sampling with a monotonic criterion.

6.1.5 Rejection Sampling in GPOKS

To achieve higher performance the rejection sampling phase in GPOKS combines the monotonic and the optimistic requirement into a single criterion. Thus, only accepting candidate functions that are both optimistic and monotonic. Figure 6.5 shows a selection of candidate functions that the GPOKS rejection sampling accept.

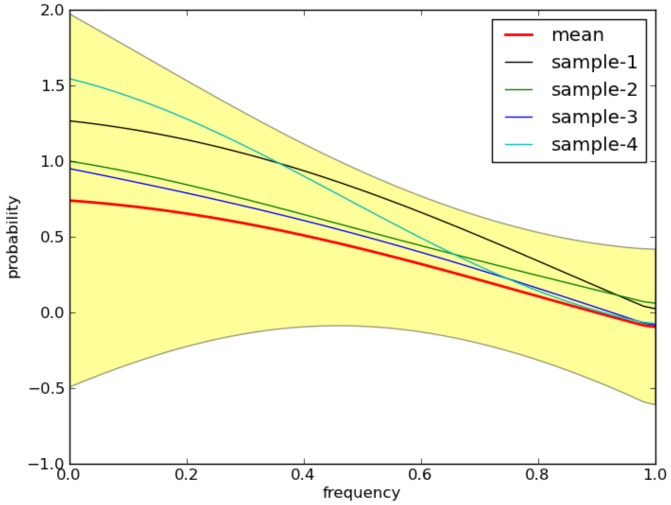


Figure 6.5: GPOKS Rejection Sampling with both monotonic and optimistic criteria.

6.2 Implementation

Armed with the conceptional understanding of GPOKS we now provide an architectural overview of our scheme in figure 6.6. As illustrated in the figure, GPOKS operate as follows:

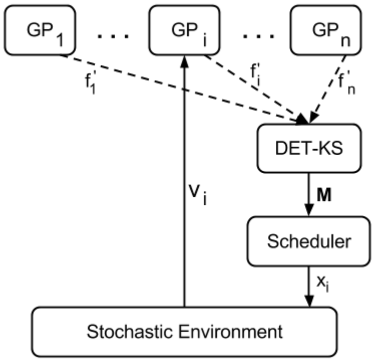


Figure 6.6: GPOKS Architectural Overview

1. A collection of GPs, one Gaussian Process GP_i , for each material i, attempts to estimate the material unit value functions, $p_i(x_i) = f_i'(x_i)$, $1 \leq i \leq n$.

2. One candidate material unit value function, $\hat{f}'_i(x_i)$, $1 \leq i \leq n$, is then sampled, using rejection sampling, from each GP_i, thus applying the TS principle of sampling functions proportionally to their likelihoods.

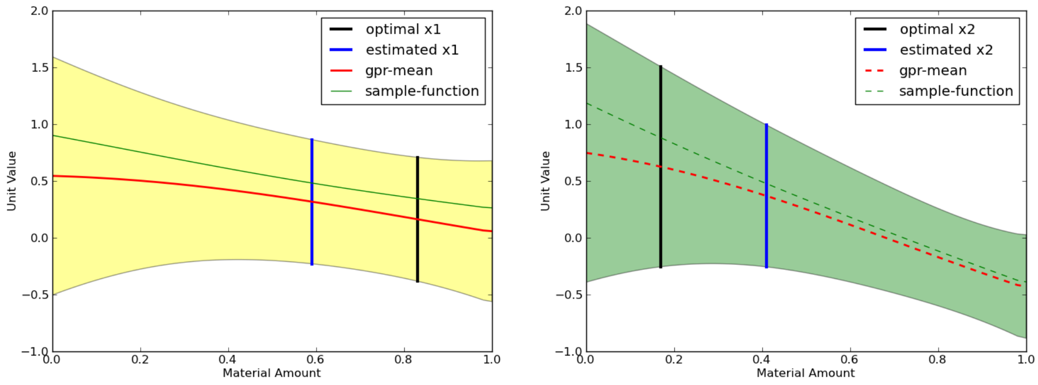
3. The *DET-KS* component in the architecture finds the optimal material mixture vector $\hat{\mathbf{M}} = [x_1, x_2, \dots, x_n]$ for the sampled material unit value functions, $\hat{f'}_i(x_i)$, $1 \leq i \leq n$, using Lagrange Multipliers.

4. One of the materials is then selected by the Scheduler component for evaluation, ensuring that each material i is selected with a frequency proportional to the amount of material x_i, assigned by $\hat{\mathbf{M}}$.

5. Finally, the *Stochastic Environment* i.e. the Stochastic NEFK, samples the true underlying probability function $p_i(x_i)$ using the selected materials *material amount* x_i, providing a feedback v_i to the corresponding GP_i, which update its Bayesian estimate of $f'_i(x_i)$.

6.3 Example Steps

To finish the section on GPOKS we will show a selection of example steps that identify the strength of the algorithm. In this context there is a total of two materials that should be *mixed*. Figure 6.7 shows the GP based estimates for the two material unit values after only 7 material value observations. It is clear from the figure that there is a *significant* uncertainty associated with the estimated material value functions, and therefore the estimated optimal material amounts $\hat{\mathbf{M}} = [\hat{x_1}, \hat{x_2}]$ are far from the optimal amounts $\mathbf{M} = [x_1, x_2]$.



Estimate of material unit value $f'_1(x_1)$ after 7 observations, with optimal and estimated material amounts x_1.

Estimate of material unit value $f'_2(x_2)$ after 7 observations, with optimal and estimated material amounts x_2.

Figure 6.7: GP_1 and GP_2 at $t = 7$

However, after 193 iterations of the GPOKS algorithm, we observe a number of fascinating properties in Figure 6.8. First of all, the Bayesian estimates of the material unit values, $f_1'(x_1)$ and $f_2'(x_2)$ have become more accurate (i.e. the Bayesian Credibility interval surrounding each point is lower). Furthermore, we observe that the estimated optimal material mixture is much closer to the optimal values. Finally, observe that the uncertainty concerning $f'(x_1)$ and $f_2'(x_2)$ varies with x_1 and x_2. The beauty of Thompson Sampling is that the exploration is guided, i.e. that exploration is performed in the areas containing potential optimal values, thus gradually zooming in on the areas that are more likely to contain the optimal material mixture.

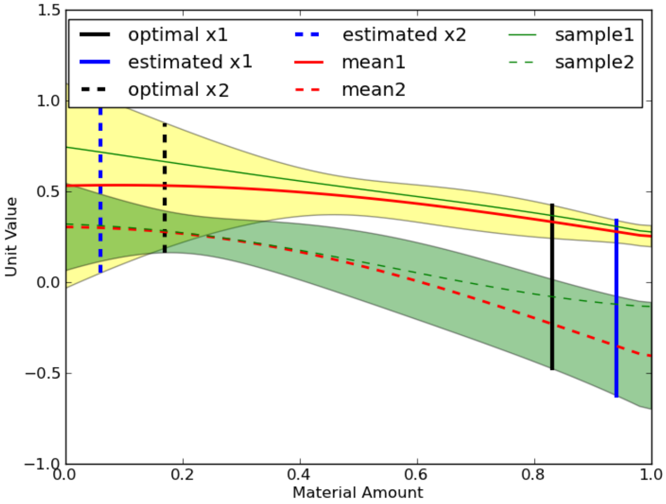


Figure 6.8: Estimate of material unit values $f_1'(x_1)$ and $f_2'(x_2)$ after 193 observations, with optimal and estimated material amounts x_1 and x_2.

6.3.1 Finite Index Set Optimization

While the above solution works, in theory, a major drawback is that GP regression is an $O(n^3)$ operation (as it requires a matrix inversion), leading for poor scalability for large time series. However, by exploiting the fact that each GP is indexed by the open interval $I_x = (0.0, 1.0)$, and the fact that the usage of a Squared Exponential Kernel gives us *very* smooth functions we can reduce n, the number of used observations in the following manner: First we convert the continuous index set into a discreet index set, consisting of k different values as $I_x \mapsto \{x_0 = 0, x_1 = \frac{1}{k-1}, x_2 = \frac{2}{k-1}, \ \ldots \ , x_k = 1\}$. The potential reduction in convergence accuracy stemming from this can be alleviated by using a higher k value[19], thus lowering the distance between the algorithms optimal point and

[19] This loss will be further explored in the experiment section later on.

the true optimal point.

Then, we recall the equations (4.5 and 4.4) for the GP regression and the SE Kernel respectively (both re-stated here to improve readability):

$$P(\mathbf{f}_*|\mathbf{y}) = \mathcal{N}(K_{*,\mathbf{f}}\ [K_{\mathbf{f},\mathbf{f}} + \sigma^2_{noise}I]^{-1}\mathbf{y},\ K_{*,*} - K_{*,\mathbf{f}}\ [K_{\mathbf{f},\mathbf{f}} + \sigma^2_{noise}I]^{-1}\ K_{\mathbf{f},*}).$$

$$K(x_p, x_q) = \sigma_f^2 \exp\left(-\frac{1}{2l^2}(x_p - x_q)^2\right) + \sigma_n^2\delta_{pq},$$

We see that the only part that affect the white noise (i.e. σ^2_{noise}) is the part of the matrix where the Kronecker delta is '1' (i.e. p equals q). Now since we are using a zero-mean prior for the GP (i.e. $m(\mathbf{x}) = \mathbf{0}$) then the distribution over each separate index x_i point, conditioned on all the observations of f_i, becomes a one-dimensional Gaussian $\sim \mathcal{N}_i\left(\bar{y_i}, \frac{\sigma^2_{noise}}{n_i}\right)$, here $\bar{y_i}$ is the average of each observation y_i at point x_i and n_i is the number of observations at x_i. This result comes from the standard Bayesian inference rules, when sampling from a Gaussian with known variance [7].

Taking this observation into account we can transform the $n \times n$ matrix $[K_{\mathbf{f},\mathbf{f}} + \sigma^2_{noise}I]$, where n is the total number of observations, into a $k \times k$ matrix (here k is the number of elements in our discrete index set). Reducing our time complexity[20] from $O(n^3)$ to $O(k)$.

In matrix notation this looks like:

$$[K_{\mathbf{f},\mathbf{f}}+\sigma^2_{noise}I] = \begin{bmatrix} K(x_0,x_0) + \frac{\sigma^2_{noise}}{n_0} & K(x_0,x_1) & \dots & K(x_0,x_k) \\ K(x_1,x_0) & K(x_0,x_1) + \frac{\sigma^2_{noise}}{n_1} & \dots & K(x_1,x_k) \\ \vdots & \vdots & \ddots & \vdots \\ K(x_k,x_0) & K(x_k,x_1) & & K(x_k,x_k) + \frac{\sigma^2_{noise}}{n_k} \end{bmatrix}$$

$$\mathbf{y} = [\bar{y_0}, \bar{y_1},\ \dots\ , \bar{y_k}]'$$

The usage of this method is exact within the resolution or accuracy of our discrete index set, as opposed to using a sparse approximation method [40] [39] that typically operates by only using a selected sub-set of the total number of observations, where the subset is based on some random sampling heuristic. These sparse methods are more suited for problems where it is infeasible to generate a discrete index set. An additional observation is that, as earlier mentioned, TS *zooms* in on the potential optimal parts requiring that a small set of observations handle the shape of the non-potential points, making it proportionally harder to select the right point to use in the sparse approximation.

[20] Note that each step still require $O(k^3)$ to perform the necessary matrix inversion, but this time does not increase with a growing number of observations, and can therefore be threated as a constant.

7 Experiment Configurations

In this section we give the experimental configuration for GPOKS following the Web Polling application outlined in the introduction. We will first give an overview of the algorithms that we use as a basis to determine the performance level of GPOKS, in addition to the configurations and settings used.

7.1 GPOKS Variations

While the GPOKS is a well defined scheme there exists several fascinating aspects of the algorithm that we can modify to provide conceptual clarity on why GPOKS work as it does and highlight possible improvements.

7.1.1 GPOKS Variants

The GPOKS Variants presented here can be divided into two groups: Those who modify the rejection sampling phase, and those who select the f_i functions in some other fashion. In total, we present seven different schemes of this type:

GPOKS:GPR – Here we use mean of the GP found using Gaussian Process Regression (GPR) as in section 4.4.2 instead of utilizing any Thompson Sampling. This is very similar to the technique found in Bayesian Optimization (BO), used in the GP-UCB algorithm covered in Section 5.3.

GPOKS:UCB – Inspired by the success of UCB-1 in the MAB context (Section 3.2.1) we adapt the GPOKS scheme to use the upper 95% confidence interval[21] instead of utilizing any TS to sample $f_i(\cdot)$.

GPOKS:IPS and **GPOKS:IPS-OTS** – Independent Point Sampling (IPS) replace the function sampling found in GPOKS with independent point sampling. This is done by generating a Gaussian distribution over each separate discrete point in the index set. And for each marginal Gaussian distribution sample a point, where f_i is then defined as the collection of these point samples. Let GPOKS:IPS-OTS denote the OTS version of this variant.

GPOKS:TS – While this variant of GPOKS utilize the regular setup found in section 6 we remove the rejection-sampling phase and instead use a *"naive"* implementation where the first sample function is used as an estimator. This is the GPOKS form closest to the "pure" Thompson Sampling algorithm.

GPOKS:MONO – The rejection sampling phase only uses the monotonic decreasing TS criteria.

[21] Since each point is marginally distributed as Gaussian Distribution we can use $\mu + 2\sigma$ to find the 95% confidence interval, cf. The 68-95-99.7 rule

GPOKS:OTS – The rejection sampling phase only uses the optimistic TS criteria.

7.2 Contending Algorithms

We can divide the contending algorithms into three different types; (1) Learning Automata based algorithms (2) Multi-Armed-Bandit Algorithms (3) Naive techniques, such as a uniform distribution.

7.2.1 Learning Automata based Solvers

LA:LAKG – Learning Automata Knapsack Game as covered in section 5.1 is used with N the number of states per LA set to 100.

LA:H-TRAA – Hierarchy of Twofold Resource Allocation Automata as covered in section 5.2 is used with N the number of states per LA set to 100, while the number of materials is restricted to a power of two, hence the material amounts used in the different experiments.

7.2.2 MAB Solvers

The Multi-Armed-Bandit solver are not in general made to handle mixture strategies but represent a definite focus of research within the reinforcement learning literature and is therefore included here. However, as outlined in Section 5.3.1 we can by using a higher number of arms represent the entire search space and thus allow MAB to *potentially* reach an optimal mixture. This is technique is used for all the MAB solver except for MAB:UCB-1-RAW who operate directly on the materials, but is in every other sense identical to MAB:UCB-1.

MAB:UCB-1 – Starts by selecting each arm once, and proceeds to select the arm with the highest confidence interval. As the number of plays grows so will the confidence intervals where we are sure of the mean value decrease, as such exploration is achieved. It also incorporates an exploration term that boost performance in areas that *potentially* could be better.

MAB:UCB-1-RAW – Identical to MAB:UCB-1 except it works directly on the materials and not on a distribution over the possible mixture values.

MAB:TS – This is the Thompson Sampling as applied to MAB problem as described in Section 3.3.1. We will use a Gaussian Distribution as our conjugate prior. The prior distribution over each arm is $\sim \mathcal{N}(0.7, 0.2$ and the likelihood variance is set as: $\sigma_{ob}^2 = 0.05$.

MAB:GP-UCB – This represent the *state-of-art* when it comes to dependent arm MAB problems as described in Section 5.3. We us a SE-Kernel with hyper-parameters:
$\{l^2 = 1.0,\ \sigma_f^2 = 1.0,\ \sigma_n^2 = 0.1\}$.

7.2.3 Naive solvers

Optimal – This plays the optimal setting found using Lagrange Multipliers.
Uniform – Plays a uniform fashion i.e. uses a equal amount of each material.

7.2.4 Multiple sites

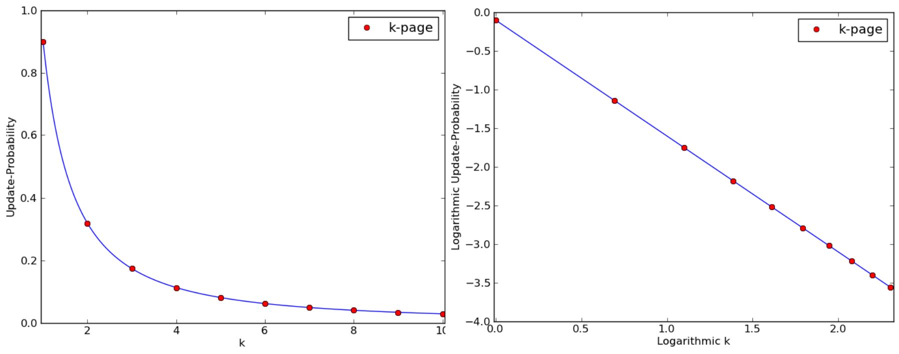
To generate realistic update probabilities for multiple web sites, a Zipf [52] like distribution is recommended [36] [51] see equation 7.1. The form of the distribution is controlled by a pair of hyper-parameters, s that control the skewness of the distribution and N that is the number of elements in the distribution. The parameter k referees to the rank of a web-page.

$$Z_k(s, N) = \frac{1/k^s}{\sum_{n=1}^{N}(1/n^s)} \tag{7.1}$$

As to easier draw comparisons to existing algorithms we a follow the lead of [17] [19] and rank our web-pages from 1 to n. Each web-page the is given a update probability from the distribution found in eq 7.2.

$$q_{(\alpha,\beta)}(k) = \frac{\alpha}{k^\beta} \tag{7.2}$$

To give some insight in how this distribution behaves we provide both a linear and a logarithmic plot for the case with 10 web pages in figure 7.1. What we can clearly see from this figure is that the update probability of the higher ranked sites gradually approaches 0 given that β is less than 1.0.



Linear scale

Logarithmic scale

Figure 7.1: Zipf distribution with $\alpha = 0.9$ and $\beta = 1.5$

The distributions of α β that we will test is given in table 7.2 below, alongside their total update probability, $\sum q_{(\alpha,\beta)}(k)$. A note of interest in this table is that not all the probability distributions sum to one, the environment $\alpha = 0.9; \beta = 1.5$ is of special interest as it is highly skewed and thus will have the largest difference between the optimal policy and the proportional one.

α	0.3	0.6	0.9
β	1.0	1.0	1.5
$\sum$	0.87	1.75	1.80

Figure 7.2: The test environments

8 Experiment Results

The following section gives the main results in this thesis. We here only report a representative sub-set of the experiments run, as the rest of the experiments show the same trend. Each configuration is run using a ensemble of 1000 independent trials and the averages (when applicable) are reported here.

8.1 Two Web Resources

This subsection we consider a scenario where only two web-pages are available for polling. We organize it as follows: First we will review the GP optimization done as part of GPOKS in Section 6.3.1 under a two material configuration. Then we report the performance characteristic of the Rejection Sampling Schemes found in GPOKS. We finish off by comparing the GPOKS Variations and the state-of-the-art algorithms. For the configuration parameters used by the different algorithms we refer to Section 7.

The main scenarios in this section is when $p_1 = 0.75, p_2 = 0.25$ since it leads to a good trade-off where it is imperative to exploit the best material, while on the other hand it is important to add a small fraction of the *sub-optimal* material 2.

8.1.1 Effective GPOKS

In Section 6.3.1 we outline a technique for exploiting the finite index set of GPOKS to increase the efficiency of the GP regression phase in GPOKS, lets call this the FAST-GPOKS, and the regular GP regression scheme (while still using a finite index set) REGULAR-GPOKS. While both algorithm versions give the exact same results, FAST-GPOKS significantly decrease the running time for longer trials, as can be seen in table 8.1. And is as such outright necessary for longer trials i.e., in a setting where the GPOKS is always running and trying to gradually improve the on the results as new observations constantly arrive. Note that the time for REGULAR-GPOKS at $t = 10000$ is missing, due to its extremely long running time (more than 10 hours).

	t=10	t=100	t=1000	t=2000	t=4000	t=10000
FAST-GPOKS	1.9s	3.1s	1.4 min	2.0 min	4 min	4 min
REGULAR-GPOKS	1.9s	3.1s	2.1 min	20.7min	2.8 hours	-
difference	0s	0s	0.7 min	18.7min	$\sim$ 2.8 hours	-

Figure 8.1: Average running time for a single trial using FAST-GPOKS and REGULAR-GPOKS. Here $p_1/p_2 = 0.75/0.25$

8.1.2 GPOKS Rejection Sampling

For the GPOKS variants we first take a look on the different ways the rejection sampling affect GPOKS. In table 8.2 one can clearly see that the usage of rejection sampling enhance the performance of GPOKS. However, what is more interesting, is the fact that both the *optimistic* and the *monotonic* variant exhibits nearly identical performance in all three configurations, leading to the conclusion that both are equally valid choices *in-average*. The performance gain by combining them into a single hybrid criteria for rejection sampling in GPOKS is minimal but noticeable, while the unmodified Thomson Sampling (GPOKS:TS) suffers from a slight loss off performance compared to the rejection sampling based variants.

Scheme	p_1/p_2	Avg[#Updates] t=10	Avg[#Updates] t=100	Avg[#Updates] t=1000
GPOKS	0.55/0.45	7.0	73.5	749.7
GPOKS:TS	0.55/0.45	5.9	68.8	738.5
GPOKS:OTS	0.55/0.45	6.1	70.4	744.4
GPOKS:MONO	0.55/0.45	6.7	72.1	746.1
GPOKS	0.75/0.25	7.3	79.8	807.5
GPOKS:TS	0.75/0.25	6.6	74.7	787.9
GPOKS:OTS	0.75/0.25	6.9	76.5	800.1
GPOKS:MONO	0.75/0.25	7.3	77.7	802.9
GPOKS	0.90/0.10	8.3	89.5	903.4
GPOKS:TS	0.90/0.10	7.8	86.6	888.8
GPOKS:OTS	0.90/0.10	8.2	88.9	900.8
GPOKS:MONO	0.90/0.10	8.3	89.2	902.3

Figure 8.2: Rejection Sampling in GPOKS for the two material case

Variance
We can also observe a big disparity in the variance for each of the different rejection sampling schemes. In table 8.3 we report the variance over the individual trial runs for each criteria, the web-polling probability used is $p_1 = 0.75$ and $p_2 = 0.25$ where each trial was performed using a 1000 iterations. Again, we see that GPOKS (with both monotonic and optimistic rejection sampling) compare favorable with the other rejection sampling schemes having the lowest variance. Surprisingly GPOKS:MONO have a observable larger variance than

GPOKS:OTS indicating that while GPOKS:MONO might deliver good results *in-average*, GPOKS:OTS will consistently deliver good results, if slightly lower than GPOKS:MONO would. Also, the clear advantage of enhancing the exploration using a rejection sampling scheme is displayed as GPOKS:TS delivers a highly variable result.

	GPOKS	GPOKS:TS	GPOKS:OTS	GPOKS:MONO
Variance	6.29	34.04	7.93	8.14

Figure 8.3: Variance in GPOKS Rejection Sampling Schemes for two materials, $p_1/p_2 = 0.75/0.25$

White-noise tolerance
While the original web-polling problem have a inherently stochastic nature, by utilizing the alternative definition of $d_i(x_i)$ found in equation: 2.3. Allowing us to test the TS schemes under varying level of white noise σ_{ws}^2 as a measurement on how robust they are. From Table 8.4 it is clear that the usage of rejection sampling increase the noise tolerance significantly, compared to the naive GPOKS:TS algorithm.

	p_1/p_2	GPOKS	GPOKS:TS	GPOKS:OTS	GPOKS:MONO
$\sigma_{ws} = 0.0$	0.75/0.25	807.5	787.9	804.1	802.9
$\sigma_{ws} = 0.2$	0.75/0.25	804.9	786.4	803.8	800.4
$\sigma_{ws} = 0.4$	0.75/0.25	802.7	772.8	797.5	799.5

Figure 8.4: White noise sensitivity in GPOKS Rejection Sampling Schemes for the two material configuration, $p_1/p_2 = 0.75/0.25$

8.1.3 GPOKS Variations

The performance of the many variations of GPOKS is found in 8.5. There are two noteworthy candidates in this table, GPOKS and GPOKS:UCB. Both delivering equal performance in terms of number of average updates found and are close to the OPTIMAL strategy.

Scheme	p_1/p_2	Avg[#Updates] t=10	Avg[#Updates] t=100	Avg[#Updates] t=1000
OPTIMAL	0.55/0.45	7.5	75.2	752.4
GPOKS	0.55/0.45	7.0	73.5	749.7
GPOKS:GPR	0.55/0.45	5.3	51.7	721.7
GPOKS:UCB	0.55/0.45	6.7	72.9	747.5
GPOKS:IPS	0.55/0.45	6.0	69.1	738.3
GPOKS:IPS-OTS	0.55/0.45	6.0	68.9	738.6
UNIFORM	0.55/0.45	7.4	74.7	747.5
OPTIMAL	0.75/0.25	8.1	81.2	812.4
GPOKS	0.75/0.25	7.3	79.8	807.5
GPOKS:GPR	0.75/0.25	6.3	73.9	798.7
GPOKS:UCB	0.75/0.25	7.4	78.7	807.4
GPOKS:IPS	0.75/0.25	5.7	72.0	775.3
GPOKS:IPS-OTS	0.75/0.25	5.5	72.6	778.4
UNIFORM	0.75/0.25	6.8	68.7	687.5
OPTIMAL	0.90/0.10	9.0	90.9	909.9
GPOKS	0.90/0.10	8.3	89.5	903.4
GPOKS:GPR	0.90/0.10	8.6	89.8	902.8
GPOKS:UCB	0.90/0.10	8.4	89.3	904.8
GPOKS:IPS	0.90/0.10	7.9	87.1	890.4
GPOKS:IPS-OTS	0.90/0.10	7.8	86.9	890.6
UNIFORM	0.90/0.10	5.9	59.0	590.0

Figure 8.5: GPOKS Variation performance

Variance
The variance of the GPOKS Variations (Table 8.6) shows that while the difference in average updates, the performance characteristics can be subtle. Here GPOKS and GPOKS:UCB shines, with their low variance, and thus their performance stability will be high, i.e. we can expect the result of using either GPOKS or GPOKS:UCB not to deviate too much from their average performance.

	GPOKS	GPOKS:GPR	GPOKS:UCB	GPOKS:IPS	GPOKS:IPS-OTS
Variance	6.29	77.66	7.93	62.91	62.30

Figure 8.6: Variance in the performance of the GPOKS Variations for the two material configuration.

White-noise tolerance
As with the GPOKS rejection sampling we test the how the GPOKS variations perform under varying level of white noise, this is found in Table 8.7. While the performance of all the algorithms decrease with a rising level of noise, GPOKS and GPOKS:UCB stands out due to their resistance to noise, here GPOKS:UCB gains a small edge on GPOKS.

	p_1/p_2	GPOKS	GPOKS:GPR	GPOKS:UCB	GPOKS:IPS-OTS	GPOKS:IPS
$\sigma_{ws} = 0.0$	0.75/0.25	807.5	798.7	807.4	778.4	775.3
$\sigma_{ws} = 0.2$	0.75/0.25	804.9	788.1	806.6	767.6	764.2
$\sigma_{ws} = 0.4$	0.75/0.25	802.7	769.4	804.0	756.3	753.4

Figure 8.7: White noise sensitivity in GPOKS Variations for the two material configuration.

8.1.4 GPOKS Contenders

The contenting algorithms to GPOKS is tested against GPOKS in this section and is reported in Table 8.8. While both LA:LAKG and LA:H-TRAA have proven capabilities, they here display a severe limit in their speed of convergence, not improving by any noticeable factor between the different configurations. This is in stark contrast to all the other algorithms that take more of an advantage of the configuration at hand. A additional contender worth noting here is the MAB:GP-UCB, delivering a performance almost equal to GPOKS, while beating both LA:LAKG and LA:H-TRAA with a good margin. Considering that MAB:GP-UCB is not constructed specifically for this application this is a very good result. It is also interesting to observe that MAB:UCB-1-RAW outperforms MAB:UCB-1.

Scheme	p_1/p_2	Avg[#Updates] t=10	Avg[#Updates] t=100	Avg[#Updates] t=1000
OPTIMAL	0.55/0.45	7.5	75.2	752.4
GPOKS	0.55/0.45	7.0	73.5	749.7
MAB:UCB-1	0.55/0.45	4.9	66.3	668.5
MAB:UCB-1-RAW	0.55/0.45	6.5	63.9	603.1
MAB:TS	0.55/0.45	6.6	66.4	709.2
MAB:GP-UCB	0.55/0.45	5.4	67.3	732.4
LA:LAKG	0.55/0.45	6.7	69.5	709.8
LA:H-TRAA	0.55/0.45	6.8	68.8	676.4
OPTIMAL	0.75/0.25	8.1	81.2	812.4
GPOKS	0.75/0.25	7.3	79.8	807.5
MAB:UCB-1	0.75/0.25	3.0	63.0	659.9
MAB:UCB-1-RAW	0.75/0.25	6.5	71.8	738.0
MAB:TS	0.75/0.25	6.3	64.1	756.7
MAB:GP-UCB	0.75/0.25	6.4	73.9	783.6
LA:LAKG	0.75/0.25	6.7	69.5	709.7
LA:H-TRAA	0.75/0.25	6.8	68.6	676.2
OPTIMAL	0.90/0.10	9.0	90.9	909.9
GPOKS	0.90/0.10	8.3	89.5	903.4
MAB:UCB-1	0.90/0.10	1.5	57.0	654.0
MAB:UCB-1-RAW	0.90/0.10	7.0	83.4	881.5
MAB:TS	0.90/0.10	5.6	60.0	842.4
MAB:GP-UCB	0.90/0.10	7.2	87.2	895.6
LA:LAKG	0.90/0.10	6.7	69.6	709.8
LA:H-TRAA	0.90/0.10	6.8	68.7	676.3

Figure 8.8: Average #updates for the GPOKS Contenders in a two material setting.

Variance
The variance in Table 8.9 should be seen in light of the performance given by the different algorithms. As such only MAB:UCB-1-RAW and LA:H-TRAA delivers results with a noteworthy high variance, the others, while still less stable than GPOKS, are *fairly* stable.

	GPOKS	MAB:UCB-1	MAB:TS	MAB:UCB-1-RAW	MAB:GP-UCB	LA:LAKG	LA:H-TRAA
Variance	6.29	13.3	52.2	162.9	62.91	2.9	651.0

Figure 8.9: Variance in the performance of the GPOKS Contenders in a two material configuration: $p_1/p_2 = 0.75/0.25$.

8.2 Multiple Web Sites

This is a multi-page web polling configuration, where a number of web sites have been generated using the procedure outlined in Section 7.2.4. Table 8.10 gives the main result for the case of 8 web-pages where, again, the token of interest is the average number of updates detected. Due to the sometimes almost flat material unit value function $f_i(x_i)$ used under some of these configurations GPOKS is only represented using GPOKS:TS and GPOKS:OTS as it is *very* time consuming to sample a monotonic *decreasing* function from a flat function.

From the Table we can see that the clear winner is GPOKS, or more specifically, it is GPOKS:UCB with GPOKS:OTS following close behind. LA:LAKG does deliver the best result for the $\alpha/\beta = 0.3/1.0$ configuration, however the poor result on the other configurations makes it loose out in the end.

Scheme	α/β	Avg[#Updates] t=10	Avg[#Updates] t=100	Avg[#Updates] t=1000
OPTIMAL	0.3/1.0	5.9	59.2	592.1
GPOKS:TS	0.3/1.0	3.2	42.1	535.2
GPOKS:OPT	0.3/1.0	3.2	43.1	545.9
GPOKS:GPR	0.3/1.0	2.6	33.2	539.0
GPOKS:UCB	0.3/1.0	2.9	45.2	563.7
LA:H-TRAA	0.3/1.0	4.9	49.6	470.0
LA:LAKG	0.3/1.0	4.9	51.9	571.8
OPTIMAL	0.6/1.0	8.7	87.1	871.7
GPOKS:TS	0.6/1.0	4.8	65.3	802.6
GPOKS:OPT	0.6/1.0	4.7	66.0	814.7
GPOKS:GPR	0.6/1.0	5.4	62.4	799.9
GPOKS:UCB	0.6/1.0	5.5	69.4	837.6
LA:H-TRAA	0.6/1.0	7.1	70.2	645.4
LA:LAKG	0.6/1.0	7.0	73.4	828.9
OPTIMAL	0.9/1.5	9.5	95.9	959.4
GPOKS:TS	0.9/1.5	5.2	76.4	894.1
GPOKS:OPT	0.9/1.5	4.9	78.9	925.0
GPOKS:GPR	0.9/1.5	8.1	90.3	926.4
GPOKS:UCB	0.9/1.5	7.3	80.6	934.5
LA:H-TRAA	0.9/1.5	6.0	60.1	566.4
LA:LAKG	0.9/1.5	6.0	64.9	855.9

Figure 8.10: Average #updates using 8 web-pages.

Variance

The variance measurement is based on the 8 pages setup, where $\alpha/\beta = 0.6/1.0$. To backup the several-order of magnitude difference between H-TRAA based algorithms we state the best and worst trial for H-TRAA: *min=396.6 max=766.7*. From Table 8.11 we can see that the usage of rejection sampling have a great effect in decreasing the variance and allows GPOKS:OTS to deliver strong and stable result. GPOKS:UCB have the most stable results, while LAKG follows on a 3_{rd} place, behind GPOKS:OTS.

	GPOKS:OTS	GPOKS:TS	GPOKS:GPR	GPOKS:UCB	LA:LAKG	LA:H-TRAA
Variance	43.9	92.5	774.3	25.7	62.91	1008.4

Figure 8.11: Variance for the different algorithms using 8 web-pages with $\alpha/\beta = 0.6/1.0$

9 Conclusions and Future Work

The stochastic non-linear fractional knapsack problem is a challenging optimization problem with numerous applications, including resource allocation. The goal is to find the most valuable mix of materials that fits within a knapsack of fixed capacity. When the value functions of the involved materials are fully known and differentiable, the most valuable mixture can be found by direct application of Lagrange multipliers. In this thesis we introduced Gaussian Process based Optimistic Knapsack Sampling (GPOKS) – a novel model based reinforcement learning scheme for solving stochastic fractional knapsack problems. The scheme is founded on Gaussian Process (GP) enabled Optimistic Thompson Sampling (OTS).

To summarize our most important empirical findings:

- Our empirical results demonstrates that the GPOKS scheme converge significantly faster than both LAKG and H-TRAA for the two material case as well as the multi-web-page configuration. This while providing more stable average results.

- We have demonstrated that Rejection Sampling, applied to OTS principle can provide a positive effect on the performance. In addition it leads to a more guided exploration lowering the variance in the result, and thus making the algorithm less sensitive to white-noise.

- By exploiting the restricted index set we where able to speed up the processing of GPOKS significantly. Ensuring that GPOKS can be considered as a practical solver when applying the SNEFK problem.

- The overall architecture of GPOKS is working flawlessly for a host of variations, where the original: model ⇒ solve ⇒ update scheme, is used a basis.

- To the best of our knowledge this is the first time the MAB:GP-UCB have been utilized for the SNEFK problem and we have shown that it is a worthy contender as a SNEFK solver, outperforming both LAKG and H-TRAA, for the two material case.

Future Work

In our further work, we will address games of interacting GPOKS for solving networked and hierarchical resource allocation problems alongside with providing a stronger theoretical background for GPOKS. We are also working on extending the GPOKS to handle resource allocation problems of a non-convex nature.

References

[1] Shipra Agrawal and Navin Goyal. Analysis of thompson sampling for the multi-armed bandit problem. In *In Conference On Learning Theory (COLT)*, COLT 2012, 2012.

[2] R. Andonov, V. Poirriez, and S. Rajopadhye. Unbounded knapsack problem: Dynamic programming revisited. *European Journal of Operational Research*, 123(2):394 – 407, 2000.

[3] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256, 2002.

[4] Peter Auer and Ronald Ortner. Ucb revisited: Improved regret bounds for the stochastic multi-armed bandit problem. *Periodica Mathematica Hungarica*, 61(1-2):55–65, 2010.

[5] Alain Berlinet and Christine Thomas-Agnan. *Reproducing kernel Hilbert spaces in probability and statistics.* Kluwer Academic Boston, 2004.

[6] PE Black. Fractional knapsack problem. *Dictionary of algorithms and data structures*, 2004.

[7] W. M. Bolstad. *Introduction to Bayesian Statistics.* Wiley, 2007.

[8] K. M. Bretthauer and B. Shetty. The Nonlinear Knapsack Problem — Algorithms and Applications. *European Journal of Operational Research*, 138:459–472, 2002.

[9] Olivier Chapelle and Lihong Li. An empirical evaluation of thompson sampling. In *Neural Information Processing Systems (NIPS)*, 2011.

[10] Nello Cristianini and John Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods.* Cambridge University Press, 2000.

[11] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification.* Wiley, New York, 2 edition, 2001.

[12] Yoav Freund and Robert E. Schapire. Game theory, on-line prediction and boosting. In *Proceedings of the ninth annual conference on Computational learning theory*, COLT '96, pages 325–332. ACM, 1996.

[13] Walter R Gilks and Pascal Wild. Adaptive rejection sampling for gibbs sampling. *Applied Statistics*, pages 337–348, 1992.

[14] S. Glimsdal and O.-C Granmo. Gaussian process based optimistic knapsack sampling with applications to stochastic resource allocation. *MAICS 2011*, 24:to appear, 2013.

[15] O.-C. Granmo. Solving two-armed bernoulli bandit problems using a bayesian learning automaton. *International Journal of Intelligent Computing and Cybernetics*, 3(2):207–234, 2010.

[16] O.-C. Granmo and B. J. Oommen. On Allocating Limited Sampling Resources Using a Learning Automata-based Solution to the Fractional Knapsack Problem. In *Proceedings of the 2006 International Intelligent Information Processing and Web Mining Conference (IIS:IIPW'06)*, Advances in Soft Computing, pages 263–272. Springer, 2006.

[17] O.-C. Granmo, B. J. Oommen, S. A. Myrer, and M. G. Olsen. Determining Optimal Polling Frequency Using a Learning Automata-based Solution to the Fractional Knapsack Problem. In *Proceedings of the 2006 IEEE International Conferences on Cybernetics & Intelligent Systems (CIS) and Robotics, Automation & Mechatronics (RAM)*, pages 73–79. IEEE, 2006.

[18] O.-C. Granmo, B. J. Oommen, S. A. Myrer, and M. G. Olsen. Learning Automata-based Solutions to the Nonlinear Fractional Knapsack Problem with Applications to Optimal Resource Allocation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(1):166–175, 2007.

[19] O.-C. Granmo and B. John Oommen. Solving Stochastic Nonlinear Resource Allocation Problems Using a Hierarchy of Twofold Resource Allocation Automata. *IEEE Transactions on Computers*, 59(4):545–560, 2010.

[20] J. Holland. *Adaptation in natural and artificial systems.* MIT Press, 1992.

[21] Emilie Kaufmann, Nathaniel Korda, and Rémi Munos. Thompson sampling: An asymptotically optimal finite-time analysis. In *Algorithmic Learning Theory*, pages 199–213. Springer, 2012.

[22] Hans Kellerer, Ulrich Pferschy, and David Pisinger. *Knapsack problems.* Springer Verlag, 2004.

[23] Levente Kocsis and Csaba Szepesvri. Bandit based monte-carlo planning. In *In: ECML-06. Number 4212 in LNCS*, pages 282–293. Springer, 2006.

[24] Ken-Li Li, Guang-Ming Dai, and Qing-hua Li. A genetic algorithm for the unbounded knapsack problem. In *Machine Learning and Cybernetics, 2003 International Conference on*, volume 3, pages 1586–1590. IEEE, 2003.

[25] Devroye Luc. Non-uniform random variate generation. *NY: Springer*, 1986.

[26] David JC MacKay. Introduction to gaussian processes. *NATO ASI Series F Computer and Systems Sciences*, 168:133–166, 1998.

[27] Silvano Martello and Paolo Toth. Upper bounds and algorithms for hard 0-1 knapsack problems. *Operations Research*, 45(5):768–778, September/October 1997.

[28] Benedict C. May, Nathan Korda, Anthony Lee, and David S. Leslie. Optimistic bayesian sampling in contextual-bandit problems. *The Journal of Machine Learning Research*, pages 2069–2106, 2012.

[29] Benedict C May and David S Leslie. Optimistic bayesian sampling in contextual-bandit problems. Technical report, Technical Report 11: 01, Statistics Group, Department of Mathematics, University of Bristol, 2011.

[30] Benedict C May and David S Leslie. Simulation studies in optimistic bayesian sampling in contextual-bandit problems. Technical report, Technical Report 11: 02, Statistics Group, Department of Mathematics, University of Bristol, 2011.

[31] Marvin Minsky and Seymour Papert. Perceptron: an introduction to computational geometry. *The MIT Press, Cambridge, expanded edition*, 19:88, 1969.

[32] Gordon E Moore. Cramming more components onto integrated circuits. *Electronics Magazine*, 1965.

[33] K. S. Narendra and M. A. L. Thathachar. *Learning Automata: An Introduction.* Prentice Hall, 1989.

[34] John Noga and Veerawan Sarbua. An online partially fractional knapsack problem. In *Proceedings of the 8th International Symposium on Parallel Architectures,Algorithms and Networks*, ISPAN '05, pages 108–112, Washington, DC, USA, 2005. IEEE Computer Society.

[35] Ole-Christoffer O.C Granmo and Stian Berg. Solving non-stationary bandit problems by random sampling from sibling kalman filters. In *Trends in Applied Intelligent Systems*, volume 6098 of *Lecture Notes in Computer Science*, pages 199–208. Springer Berlin Heidelberg, 2010.

[36] S. Pandey, K. Ramamritham, and S. Chakrabarti. Monitoring the Dynamic Web to Respond to Continuous Queries. In *12th International World Wide Web Conference*, pages 659–668. ACM Press, 2003.

[37] D. Pisinger. A minimal algorithm for the 0-1 knapsack problem. *Operations Research*, 45(5):758–767, September/October 1997.

[38] George Pólya. *How to Solve It.* Princeton University Press, Connecticut, USA, 1957.

[39] Joaquin Quiñonero Candela and Carl Edward Rasmussen. A unifying view of sparse approximate gaussian process regression. *J. Mach. Learn. Res.*, 6:1939–1959, December 2005.

[40] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning.* The MIT Press, 2006.

[41] H Robbins. *Some aspects of the sequential design of experiments.* Bulletin of the merican Mathematical Society 55, 1952.

[42] Frank Rosenblatt. The perceptron. *Psych. Rev*, 65(6):386–408, 1958.

[43] Paat Rusmevichientong and David P. Williamson. An adaptive algorithm for selecting profitable keywords for search-based advertising services. In *Proceedings of the 7th ACM conference on Electronic commerce*, EC '06, pages 260–269. ACM, 2006.

[44] D. Pisinger S. Martello and P. Toth. New trends in exact algorithms for the 01 knapsack problem. *European Journal of Operational Research*, 123(2):325 – 332, 2000.

[45] Matthias Seeger. Gaussian processes for machine learning. *International Journal of Neural Systems*, 14(02):69–106, 2004.

[46] Niranjan Srinivas, Andreas Krause, Sham M. Kakade, and Matthias Seeger. Gaussian process bandits without regret: An experimental design approach. *CoRR*, abs/0912.3995, 2009.

[47] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction.* MIT Press, 1998.

[48] David Silver Sylvain Gelly. Achieving master level play in 9 x 9 computer go. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 1537–1540. AAAI, 2008.

[49] Yizao Wang Sylvain Gelly. Exploration exploitation in go: Uct for monte-carlo go. In *Advances in Neural Information Processing Systems 19.* NIPS, 2006.

[50] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[51] J. L. Wolf, M. S. Squillante, J. Sethuraman, and K. Ozsen. Optimal Crawling Strategies for Web Search Engines. In *11th International World Wide Web Conference*, pages 136–147. ACM Press, 2002.

[52] G. Zipf. *Human Behavior and the Principle of Least Effort.* Addison-Wesley, 1949.

A DET-KS Proof

The DET-KS algorithm in section 6.1.1 is proved in two steps. First a simpler problem is presented and solved in a obvious optimal manner. Then DET-KS is mapped onto this simple algorithm, and the result follows. We have taken the liberty of restating the DET-KS algorithm here for readability.

```
Algorithm: DET-KS
Input: Set of functions $\mathbf{f} = \{f_1, \ldots, f_n\}$ and $c$ the knapsack capacity.
Initialization: $\mathbf{M}[1] = \ldots = \mathbf{M}[n] = \epsilon$;
# Typically $\epsilon$ can be set to some sufficiently small value.
Method:
While $sum(\mathbf{M}) =< c$ Do
    1. Find the material $i$ that have the smallest delta value weighted by its
       mixture:
       $i = \operatorname*{argmin}_{j \in \{1,n\}} \; [\mathbf{M}[j] \times f(\mathbf{M}[j])] - [(\mathbf{M}[j] + \epsilon) \times f(\mathbf{M}[j] + \epsilon)].$
    2. Increase the amount of $f_i$ in the mixture: $\mathbf{M}[i] = \mathbf{M}[i] + \epsilon$.
End While
Return M
End Algorithm: DET-KS
```

Proof:
Imagine a rich but stingy man; let's call him Scrooge, who has money lying around in bags of assorted size. He has been made to promise away a fixed amount of moneybags, but wants to give away as little as possible. The obvious optimal way of doing this, is to give away the smallest bags. His strategy for picking bags is to iteratively pick the smallest remaining bag until he has picked the requisite number of bags. It is also possible to see that this algorithm is optimal in that it produces the smallest bags.

Now, we complicate this problem a little bit, by sorting the bags into piles sorted by increasing bag size. Again, we see that if the rich man wants to give away as little as possible, an optimal pick strategy will be to iteratively see which pile has the smallest bag at the top, and then to pick this smallest bag. Let's call this *Scrooge's algorithm.*

In the DET-KS setting, we are dealing with concave functions [22] f_i, and we want to minimize $\sum_i k_i\epsilon \times f_i(k_i\epsilon)$ where the sum of the non-negative integers k_i is a fixed constant c. For each i, the concavity property means that $\Delta_{i,m} = \Delta f_i(m\epsilon) = (m-1)\epsilon f_i((m-1)\epsilon) - m\epsilon f_i(m\epsilon)$ is an increasing sequence.

[22] Note that f_i keeps it concave property when scaled by an arbitrary positive scalar

We now translate this into the Scrooge setting, by letting each $\Delta_{i,m}$ corresponds to a money bag to be given away, and each function f_i to a pile of such $\Delta_{i,m}$ sorted by increasing value of the parameter m, this holds due to the concave property of f_i. The optimal solution remains to iteratively give away the smallest money bag ($\Delta_{i,m}$), and an optimal algorithm for this is *Scrooge's algorithm.* □

B Short Paper

We here include the paper that was presented at MAICS 2013 as part of the master thesis process.

Gaussian Process Based Optimistic Knapsack Sampling with Applications to Stochastic Resource Allocation

Sondre Glimsdal and Ole-Christoffer Granmo
Department of ICT
University of Agder
Norway
{sondre.glimsdal, ole.granmo}@uia.no

Abstract

The stochastic non-linear fractional knapsack problem is a challenging optimization problem with numerous applications, including resource allocation. The goal is to find the most valuable mix of materials that fits within a knapsack of fixed capacity. When the value functions of the involved materials are fully known and differentiable, the most valuable mixture can be found by direct application of Lagrange multipliers. However, in many real-world applications, such as web polling, information about material value is uncertain, and in many cases missing altogether. Surprisingly, without prior information about material value, the recently proposed Learning Automata Knapsack Game (LAKG) offers arbitrarily accurate convergence towards the optimal solution, simply by interacting with the knapsack on-line.

This paper introduces Gaussian Process based Optimistic Knapsack Sampling (GPOKS) — a novel model-based reinforcement learning scheme for solving stochastic fractional knapsack problems, founded on Gaussian Process (GP) enabled Optimistic Thompson Sampling (OTS). Not only does this scheme converge significantly faster than LAKG, GPOKS also incorporates GP based learning of the material values themselves, forming the basis for OTS supported balancing between exploration and exploitation. Using resource allocation in web polling as a proof-of-concept application, our empirical results show that GPOKS consistently outperforms LAKG, the current top-performer, under a wide variety of parameter settings.

1 Introduction

The Internet can be seen as a massive collection of ever-changing information, continuously evolving as web resources are created, edited, deleted, and replaced (Pandey, Ramamritham, & Chakrabarti 2003). Obtaining adequate information from the Internet is crucial for many tasks, including social media analytics, counter terrorism, and business intelligence. It is thus important that the applied search engines and web-monitoring frameworks are able to keep their indexes and caches complete and up-to-date. Achieving this, of course, relies on detecting the changes that the web resources undergo, typically by means of polling.

The problem of balancing polling capacity optimally among web resources, with limited prior information, was essentially unsolved until the Learning Automata Knapsack Game (LAKG) was introduced in 2006 as a generic and adaptive solution to the so-called *Stochastic Non-linear Equality Fractional Knapsack (NEFK) Problem* (Granmo *et al.* 2006). Before that, the simplest and perhaps most common polling approach was to allocate the available polling capacity uniformly among the web resources being monitored, polling them all with the same fixed frequency, constrained by the available polling capacity. This uniform polling strategy is clearly sub-optimal since web resources evolve at different speed. For slowly changing web resources, a high polling frequency translates into a correspondingly large number of unfruitful polls. Conversely, for quickly evolving web resources, a too low polling frequency leads to potential loss of information or acting on out-dated information. In brief, without balancing the allocation of the available polling capacity, wasting resources polling one resource may in turn prevent us from polling another more attractive resource, thus degrading overall performance.

A two phase strategy has been proposed to address the latter inefficiency: In the first phase, the uniform strategy is applied, which allows the update probability of monitored web resources to be estimated. By treating these probability estimates as the true ones, Lagrange multipliers can be applied to find an allocation of capacity that is optimal for the *estimated* values (Pandey, Ramamritham, & Chakrabarti 2003). However, this method needs an arbitrary long estimation phase to approach the optimal solution in the second phase. That is, one either has to accept a sub-optimal final solution because the update probability estimates are inaccurate, or one must wait an extensive amount of time till the estimates have become sufficiently accurate, allowing a better solution in the second phase. Also note that evolving update probabilities render the solution found with the latter approach progressively more inaccurate.

This paper introduces Gaussian Process based Optimistic Knapsack Sampling (GPOKS) — a novel scheme for solving stochastic knapsack problems founded on Gaussian Process (GP) (Rasmussen & Williams 2006) based Thompson Sampling (TS) (Thompson 1933; Granmo 2010), enhanced by the principles of *Optimistic* TS (May *et al.* 2012). As we shall see, not only does this scheme converge significantly faster than LAKG, GPOKS also incorporates GP based learning of the material unit values themselves, forming the basis for TS based exploration and exploitation. This allows GPOKS to gradually shift from estimation to optimization, starting with pure estimation and converging towards pure optimization.

In (Granmo 2010) we reported a *Bayesian* technique for solving bandit like problems, revisiting the *Thompson Sampling* (Thompson 1933) principle pioneered in 1933. This revisit lead to novel schemes for handling multi-armed and dynamic (restless) bandit problems (Granmo & Berg 2010; Gupta, Granmo, & Agrawala 2011a; 2011b), and empirical results demonstrated the advantages of these techniques over established top performers. Furthermore, we provided theoretical results stating that the original technique is instantaneously self-correcting and that it converges to only pulling the optimal arm with probability as close to unity as desired. We now expand this principle to support Thompson Sampling for Stochastic NEFK Problems.

1.1 Formal Problem Formulation

In order to appreciate the qualities of the Stochastic NEFK Problem, it is beneficial to view the problem in light of the classical *linear* Fractional Knapsack (FK) Problem. Indeed, the Stochastic NEFK Problem generalizes the latter problem in two significant ways. Both of the two problems are *briefly* defined below.

The Linear Fractional Knapsack (FK) Problem: The linear FK problem is a classical continuous optimization problem which also has applications within the field of resource allocation. The problem involves n materials of different value v_i per unit volume, $1 \leq i \leq n$, where each material is available in a certain amount $x_i \leq b_i$. Let $f_i(x_i)$ denote the value of the amount x_i of material i, i.e., $f_i(x_i) = v_i x_i$. The problem is to fill a knapsack of fixed volume c with the material mix $\vec{x} = [x_1, \ldots, x_n]$ of maximal value $\sum_1^n f_i(x_i)$ (Black 2004).

The Nonlinear Equality FK (NEFK) Problem: One important extension of the above classical problem is the *Nonlinear Equality* FK problem with a separable and concave objective function. The problem can be stated as follows (Kellerer, Pferschy, & Pisinger 2004):

$$\begin{array}{ll} \text{maximize} & f(\vec{x}) = \sum_1^n f_i(x_i) \\ \text{subject to} & \sum_1^n x_i = c \text{ and } \forall i \in \{1, \ldots, n\}, x_i \geq 0. \end{array}$$

Since the objective function is considered to be concave, the value function $f_i(x_i)$ of each material is also concave. This means that the derivatives of the material value functions $f_i(x_i)$ with respect to x_i, (hereafter denoted f_i'), are non-increasing. In other words, the material value *per unit volume* is no longer constant as in the linear case, but decreases with the material amount, and so the optimization problem becomes:

$$\begin{array}{ll} \text{maximize} & f(\vec{x}) = \sum_1^n f_i(x_i), \\ & \quad \text{where } f_i(x_i) = \int_0^{x_i} f_i'(x_i) dx_i \\ \text{subject to} & \sum_1^n x_i = c \text{ and } \forall i \in \{1, \ldots, n\}, x_i \geq 0. \end{array}$$

Efficient solutions to the latter problem, based on the principle of Lagrange multipliers, have been devised. In short, the optimal value occurs when the derivatives f_i' of the material value functions are equal, subject to the knapsack constraints (Bretthauer & Shetty 2002):

$$\begin{array}{l} f_1'(x_1) = \cdots = f_n'(x_n) \\ \sum_1^n x_i = c \text{ and } \forall i \in \{1, \ldots, n\}, x_i \geq 0. \end{array}$$

The Stochastic NEFK Problem: In this paper we generalize the above nonlinear equality knapsack problem. First of all, we let the material value per unit volume for any x_i be a *probability* function $p_i(x_i)$. Furthermore, we consider the distribution of $p_i(x_i)$ to be *unknown*. That is, each time an amount x_i of material i is placed in the knapsack, we are only allowed to observe an instantiation of $p_i(x_i)$ at x_i, and not $p_i(x_i)$ itself.[1] Given this stochastic environment, we intend to devise an on-line incremental scheme that learns the mix of materials of maximal *expected* value, through a series of informed guesses. Thus, to clarify issues, we are provided with a knapsack of fixed volume c, which is to be filled with a mix of n different materials. However, unlike the NEFK, in the Stochastic NEFK Problem the unit volume value of a material i, $1 \leq i \leq n$, is a random quantity — it takes the value 1 with probability $p_i(x_i)$ and the value 0 with probability $1 - p_i(x_i)$, respectively. As an additional complication, $p_i(x_i)$ is nonlinear in the sense that it decreases monotonically with x_i, i.e., $x_{i_1} \leq x_{i_2} \Leftrightarrow p_i(x_{i_1}) \geq p_i(x_{i_2})$.

Since unit volume values are random, we operate with expected unit volume values rather than the actual unit volume values. With this understanding, and the above perspective in mind, the expected value of the amount x_i of material i, $1 \leq i \leq n$, becomes $f_i(x_i) = \int_0^{x_i} p_i(u) du$. Accordingly, the expected value per unit volume[2] of material i becomes $f_i'(x_i) = p_i(x_i)$. In this stochastic and non-linear version of the FK problem, the goal is to fill the knapsack so that the expected value $f(\vec{x}) = \sum_1^n f_i(x_i)$ of the material mix contained in the knapsack is maximized. Thus, we aim to:

$$\begin{array}{ll} \text{maximize} & f(\vec{x}) = \sum_1^n f_i(x_i), \\ & \text{where } f_i(x_i) = \int_0^{x_i} p_i(u) du, p_i(x_i) = f_i'(x_i) \\ \text{subject to} & \sum_1^n x_i = c \text{ and } \forall i \in \{1, \ldots, n\}, x_i \geq 0. \end{array}$$

A fascinating property of the above problem is that the amount of information available to the decision maker is limited — the decision maker is only allowed to observe the current unit value of each material (either 0 or 1). That is, each time a material mix is placed in the knapsack, the unit value of each material is provided to the decision maker. The actual outcome probabilities $p_i(x_i)$, $1 \leq i \leq n$, however, remain *unknown*. As a result of the latter, the expected value of the material mix must be maximized by means of trial-and-error, i.e., by experimenting with different material mixes and by observing the resulting random unit value outcomes.

[1] For the sake of consistency with previous work on the Stochastic NEFK Problem, we here model stochastic material unit values using Bernoulli trials. However, since GPOKS is based on Gaussian Processes, the central limit theorem opens up for addressing a number of other distributions too. Furthermore, there exist dedicated kernel functions for a variety of distributions.

[2] We hereafter use $f_i'(x_i)$ to denote the derivative of the expected value function $f_i(x_i)$ with respect to x_i.

1.2 Paper Contributions

The contributions of this paper can be summarized as follows:

1. We combine Bayesian modeling with reinforcement learning to provide a novel solution to the Stochastic NEFK Problem.
2. We propose the first reinforcement learning scheme that combines Gaussian Processes (Rasmussen & Williams 2006) with Thompson Sampling (Thompson 1933; Granmo 2010).
3. We introduce GP based sampling mechanisms in the spirit of Optimistic Thompson Sampling (May *et al.* 2012) for increased performance.
4. The resulting scheme persistently outperforms state-of-the-art approaches when applied to resource allocation in web polling.

These contributions form the first steps towards establishing a new family of reinforcement learning schemes that provide on-line solutions to stochastic versions of classical optimization problems. This is achieved by carefully designing Bayesian models that capture the nature of the optimization problems, applying TS principles to address the exploration/exploitation dilemma in on-line learning and control.

1.3 Paper Outline

In Section 2, we present our scheme for Gaussian Process Based Optimistic Knapsack Sampling (GPOKS). We start with a brief introduction to Gaussian Processes before we propose how Gaussian Processes can enable Thompson Sampling — the current leader when it comes to solving Bernoulli Bandit Problems (Granmo 2010) — for exploration and exploitation when solving on-line Stochastic NEFK problems. Then, in Section 3, we define the web resource allocation polling problem in more detail, following up with an evaluation of GPOKS compared with state-of-the-art. We conclude in Section 4 and present pointers for further work.

2 Gaussian Process Based Optimistic Knapsack Sampling (GPOKS)

The conflict between exploration and exploitation is a well-known problem in reinforcement learning, and other areas of artificial intelligence. The multi-armed bandit problem captures the essence of this conflict, and has thus occupied researchers for over fifty years (Wyatt 1997). In brief, an agent sequentially pulls one of multiple arms attached to a gambling machine, with each pull resulting in a random reward. The reward distributions are unknown, and thus, one must balance between exploiting existing knowledge about the arms, and obtaining new information.

We are here facing a similar problem, however, instead of seeking the singly best material (bandit arm), we need to find a mixture of materials, also referred to as a *mixed strategy* in Game Theory. Recently, GP optimization has been addressed from a bandit problem perspective (Srinivas N. & M. 2010), allowing the GP to be explored globally with as few evaluations as possible based on so-called upper confidence bounds. Inspired by the success of GP based optimization, we here propose a novel GP based model for stochastic NEFK problems, where a *collection* of GPs captures the individual material unit values. Based on the GP colletion, Thompson Sampling is applied to sample likely deterministic NEFK problem instances from the GPs. These, in turn, are solved based on Lagrange Multipliers, producing a *potential* solution to the problem at hand.

2.1 Gaussian Processes based Representation of Material Unit Value

A Gaussian Process (GP) is a stochastic process that represents a function as a multivariate Gaussian distribution (Rasmussen & Williams 2006). It is specified as a tuple $\mathcal{GP} = (\mu(\vec{x}), K(\cdot,\cdot))$ where $\mu(\cdot)$ is the mean function, typically assigned $\mu(\vec{x}) = \vec{0}$, and $K(\cdot,\cdot)$ is a kernel that specifies the covariance matrix for the random vector $\vec{x}$. In this paper, we use the one dimensional *Squared Exponential* kernel (eq. 1), configured by the hyper parameters $\vec{\theta} = \{l, \sigma_f^2, \sigma_n^2\}$.

$$K(x_p, x_q) = \sigma_f^2 \exp(-\frac{1}{2l^2}(x_p - x_q)^2)) + \sigma_n^2 \delta_{pq} \qquad (1)$$

Here l is the characteristic length-scale parameter that determines how rapidly the correlation should decay as the distance between x_p and x_q increases, σ_f^2 is the signal variance and σ_n^2 is white noise (note that δ_{pq} here denotes the Kronecker delta between x_p and x_q). For further information on GPs we refer to (Rasmussen & Williams 2006).

By way of example, Figure 1 illustrates how the posterior distribution over possible material unit value functions for a given material i can be represented by means of a GP. The x-axis measures the amount of material, x_i, while the y-axis provides the material unit value $f_i'(x_i)$. The mean and 95% confidence interval is included, as well as four samples indicating possible candidates for $f_i'(x_i)$. Note that since the Stochastic NEFK problem deals with non-increasing unit value functions, $f_i'(x_i)$, we apply Rejection Sampling to sample from the distribution of non-increasing functions. Similarly, "optimistic" sampling, as pioneered by May et al. (May *et al.* 2012), is realized by rejecting sampled functions that drop below the estimated mean.

2.2 Architectural Overview of GPOKS

Figure 2 provides an architectural overview of our scheme. As illustrated in the figure, GPOKS operates as follows:

1. A collection of GPs, one Gaussian Process, GP$_i$, for each material i, attempts to estimate the material unit value functions, $f_i'(x_i), 1 \leq i \leq n$.
2. One candidate material unit value function, $\hat{f}_i'(x_i), 1 \leq i \leq n$, is then sampled from each GP$_i$, thus applying the TS principle of sampling functions proportionally to their likelihoods.
3. The *DET-KS* component in the architectue finds the optimal material mixture $\hat{\mathbf{M}} = [\mathbf{x_1}, \ldots, \mathbf{x_n}]$ for the sampled material unit value functions, $\hat{f}_i'(x_i), 1 \leq i \leq n$, using Lagrange multipliers.

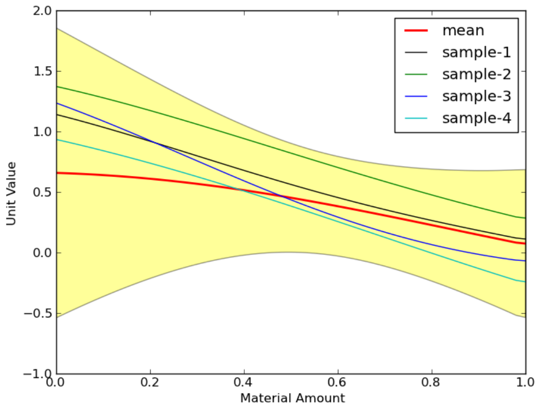


Figure 1: Gaussian Process based representation of material unit value

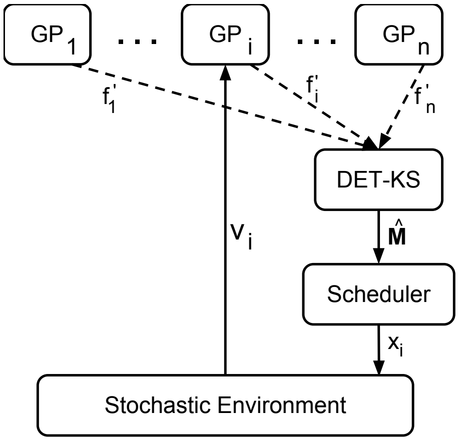


Figure 2: GPOKS Architectural Overview

4. One of the materials is then selected by the *Scheduler* component for evaluation, ensuring that each material i is selected with a frequency that is proportional to the amount of material, x_i, assigned by $\hat{\mathbf{M}}$.
5. Finally, the *Stochastic Environment*, i.e., the Stochastic NEFK, samples the true outcome probability function, $p_i(x_i)$, at x_i, providing feedback v_i to the corresponding GP$_i$, which updates its Bayesian estimate of $f_i'(x_i)$.

By following the above steps our goal is to gradually improve our "best guesses" so that each iteration successively brings us closer to the optimal solution of the targeted Stochastic NEFK problem.

2.3 Example Steps

Figure 3 and 4 show the GP based estimates for the unit value of two materials, $f_1'(x_1)$ and $f_2'(x_2)$, after only 5 material value observations. As can be seen, uncertainty about the material unit value functions is significant, and the estimated optimal material amounts $\hat{M} = [\hat{x}_1, \hat{x}_2]$ are far from the optimal amounts $M = [x_1, x_2]$.

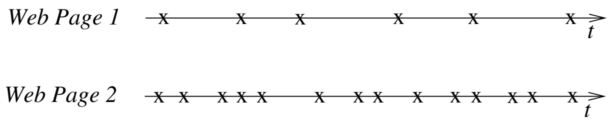


Figure 6: Web resource changes occurring over time. An 'x' on the time-lines denotes that the respective web resource has changed.

However, after 193 iterations of the GPOKS algorithm, we observe a number of fascinating properties in Figure 5. First of all, the Bayesian estimates of the material unit values, $f_1'(x_1)$ and $f_2'(x_2)$, have become more accurate. Furthermore, we observe that the estimated optimal material mixture is now much closer to the optimal mixture. Finally, observe that the uncertainty concerning $f_1'(x_1)$ and $f_2'(x_2)$ varies with x_1 and x_2. The beauty of Thompson Sampling is that the observations are collected with gradually increasing exploitation, zooming in on the areas that are most likely to contain the optimal material mixture.

3 Application: Web Polling

Having obtained a solution to the model in which we set the NEFK, we shall now demonstrate how we can utilize this solution for the current problem being studied, namely, the optimal web-polling problem.

Web resource monitoring consists of repeatedly polling a selection of web resources so that the user can detect changes that occur over time. Clearly, as this task can be prohibitively expensive, in practical applications, the system imposes a constraint on the *maximum* number of web resources that can be polled per time unit. This bound is dictated by the governing communication bandwidth, and by the speed limitations associated with the processing. Since only a fraction of the web resources can be polled within a given unit of time, the problem which the system's analyst encounters is one of determining which web resources are to be polled. In such cases, a reasonable choice of action is to choose web resources in a manner that maximizes the number of changes detected, and the optimal allocation of the resources involves trial-and-error. As illustrated in Figure 6, web resources may change with varying frequencies (that are unknown to the decision maker), and changes appear more or less randomly. Furthermore, as argued elsewhere, (Granmo & Oommen 2006; Granmo *et al.* 2006; 2007), the probability that an individual web resource poll uncovers a change on its own decreases monotonically with the polling frequency used for that web resource.

Although several nonlinear criterion functions for measuring web monitoring performance have been proposed in the literature (e.g., see (Pandey, Ramamritham, & Chakrabarti 2003; Wolf *et al.* 2002)), from a broader viewpoint they are mainly built around the basic concept of *update detection probability*, i.e., the probability that polling a web resource results in new information being discovered. Therefore, for the purpose of conceptual clarity, we will use

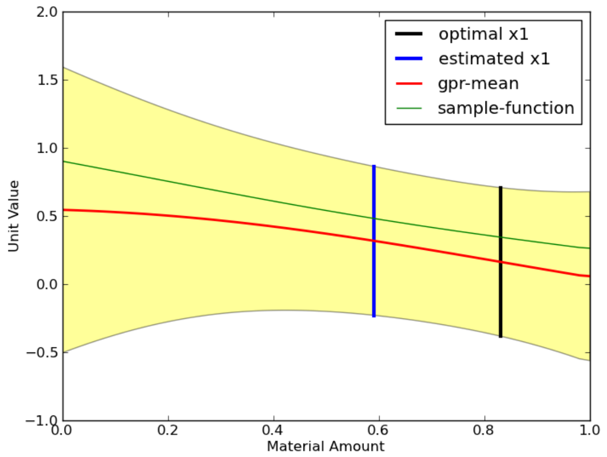


Figure 3: Estimate of material unit value $f_1'(x_1)$ after 7 observations, with optimal and estimated material amounts x_1.

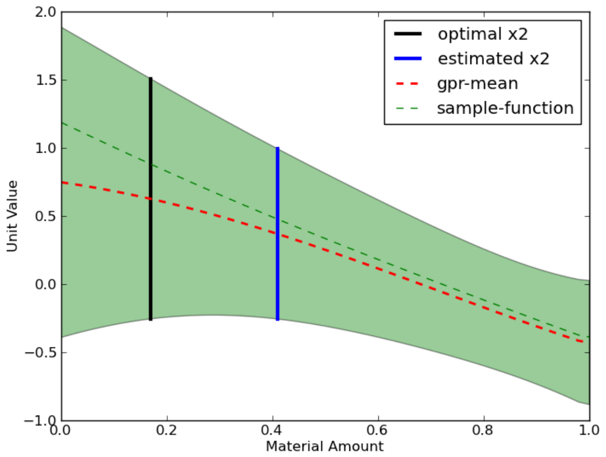


Figure 4: Estimate of material unit value $f_2'(x_2)$ after 7 observations, with optimal and estimated material amounts x_2.

the update detection probability as the token of interest in this paper. To further define our notion of web monitoring performance, we consider that time is discrete with the time interval length T to be the atomic unit of decision making. In each time interval every single web resource i has a constant probability q_i of remaining *unchanged*. Furthermore, when a web resource is updated/changed, the update is available for detection only until the web resource is updated again. After that, the original update is considered lost. For instance, each time a newspaper web resource is updated, previous news items are replaced by the most recent ones.

In the following, we will denote the update detection probability of a web resource i as d_i. Under the above conditions, d_i depends on the frequency, x_i, that the resource is polled with, and is modeled using the following expression:

$$d_i(x_i) = 1 - q_i^{\frac{1}{x_i}}.$$

By way of example, consider the scenario that a web resource remains unchanged in any single time step with probability 0.5. Then polling the web resource uncovers new information with probability $1 - 0.5^3 = 0.875$ if the web resource is polled every 3^{rd} time step (i.e., with frequency $\frac{1}{3}$) and $1 - 0.5^2 = 0.75$ if the web resource is polled every 2^{nd} time step. As seen, increasing the polling frequency reduces the probability of discovering new information on each polling.

Given the above considerations, our aim is to find the resource polling frequencies $\vec{x}$ that maximize the expected number of pollings uncovering new information per time step:

$$\begin{array}{ll} \text{maximize} & \sum_1^n x_i \times d_i(x_i) \\ \text{subject to} & \sum_1^n x_i = c \text{ and } \forall i = 1, \ldots, n, x_i \geq 0. \end{array}$$

3.1 GPOKS Solution

In order to find a solution to the above problem we must define the Stochastic Environment that GPOKS is to interact with. As seen in Section 2, the Stochastic Environment consists of the unit volume value functions $\{f_1'(x_1), f_2'(x_2), \ldots, f_n'(x_n)\}$, which are unknown to GPOKS. We identify the nature of these functions by applying the principle of Lagrange multipliers to the above maximization problem. In short, after some simplification, it can be seen that the following conditions characterize the optimal solution:

$$\begin{array}{l} d_1(x_1) = d_2(x_2) = \cdots = d_n(x_n) \\ \sum_1^n x_i = c \text{ and } \forall i = 1, \ldots, n, x_i \geq 0. \end{array}$$

Since we are not able to observe $d_i(x_i)$ or q_i directly, we base our definition of $\{f_1'(x_1), f_2'(x_2), \ldots, f_n'(x_n)\}$ on the result of polling web resources. Briefly stated, we want $f_i'(x_i)$ to instantiate to the value 0 with probability $1 - d_i(x_i)$ and to the value 1 with probability $d_i(x_i)$. Accordingly, if the web resource i is polled and i has been updated since our last polling, then we consider $f_i'(x_i)$ to have been instantiated to 1. And, if the web resource i is unchanged, we consider $f_i'(x_i)$ to have been instantiated to 0.

3.2 Empirical Results

In this section we evaluate GPOKS and compare its performance with the currently best performing algorithm, LAKG. While H-TRAA possesses better scalability than LAKG (Granmo & Oommen 2010), for two material problems, their performance is identical because the hierarchical setup of H-TRAA does not come into play. For clarification we will also include some promising variants of GPOKS. Here follows an overview of a selection of the policies that we have investigated:

Uniform: The uniform policy allocates monitoring resources uniformly across all web resources. This classical

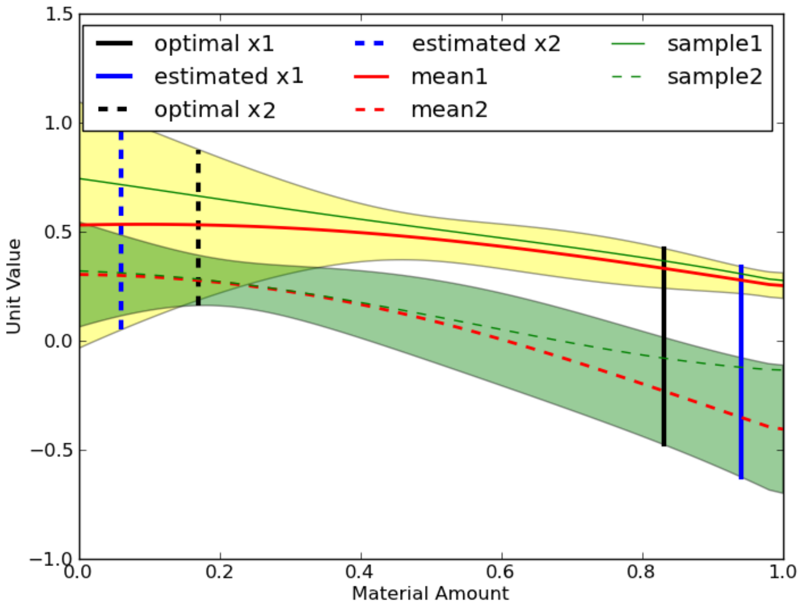


Figure 5: Estimate of material unit values $f'_1(x_1)$ and $f'_2(x_2)$ after 193 observations, with optimal and estimated material amounts x_1 and x_2.

policy can, of course, be applied directly in an unknown environment.

LAKG: The LAKG scheme is basically a game between so-called Learning Automata (Narendra & Thathachar 1989). They start off from a uniform policy and gradually improves toward the optimal configuration through a sequence of small jumps across a discretized search space. In all our experiments the resolution of LAKG is set to 100 states.

Optimal: This policy requires that update frequencies are known, and finds the optimal solution based on the principle of Lagrange multipliers (Pandey, Ramamritham, & Chakrabarti 2003; Wolf *et al.* 2002).

GPOKS - Mean: To highlight the advantage of our Optimistic Thompson Sampling approach, we also test a simpler scheme where we use the mean of the GPs when estimating the optimal solution rather than sampling functions from the GPs.

We have conducted numerous experiments using various configurations, such as different noise parameters and update probabilities. Here, we present a representative subset of these, as they all show the same trend. Performance is measured as the average *accumulated* number of web resource updates found.

For these experiments, we used an ensemble of 1000 independent replications, each random generator seeded with a unique number, to maximize the precision of the reported results. In order to provide a robust overview of the performance of GPOKS, we investigated three radically different update probability configurations for web resource pairs. In the first one, $q_1 = 0.9/q_2 = 0.1$, one web resource is updated significantly more often than the other. A more moderate version of the latter configuration, $q_1 = 0.75/q_2 = 0.25$, was also investigated. Furthermore, we measured performance when the two web resources have almost equal update probability, $q_1 = 0.55/q_2 = 0.45$. Finally, we also investigated the robustness of GPOKS by adding increasing amount of white-noise, (w_σ), to the feedback given to GPOKS. Note that, for the sake of fairness, we applied the same kernel hyper-parameters, $\theta = \{1.0, 1.0, 0.1\}$, for all the GP based strategies, without further optimization.

Table 1 reports the performance of the different policies[3]. As can be seen, GPOKS clearly outperforms LAKG when facing the $q_1 = 0.9/q_2 = 0.1$ configuration, with GPOKS detecting on average approximately 8 more updates than LAKG over 1000 time steps. Also note how remarkably close GPOKS gets to the optimal performance, missing on average merely 7 web resource updates over 1000 time steps. We observe similar results for the $q_1 = 0.75/q_2 = 0.25$ configuration. Finally, for the $q_1 = 0.55/q_2 = 0.45$ configuration, we observe that the performance of LAKG and GPOKS becomes more similar. This can be explained by the prior bias of LAKG, starting from a uniform allocation of resources. This gives LAKG an advantage over GPOKS, which are largely unbiased when it comes to prior belief about update probabilities. Finally, notice the performance loss caused by using the mean of the GPs (GPOKS-Mean) instead of TS. This trend is further explored in Ta-

[3] Note that all of the setups apply a small degree of white noise $(w_\sigma = 0.1)$.

ble 2, where we increase the amount of white noise affecting feedback. We then observe that GPOKS is surprisingly robust towards noisy feedback compared to GPOKS-Mean. This can be explained by the greedy nature of GPOKS-Mean, which is less inclined to explore the space of functions encompassed by the GPs, thus being more easily mislead by noise.

4 Conclusions and Further Work

The stochastic non-linear fractional knapsack problem is a challenging optimization problem with numerous applications, including resource allocation. The goal is to find the most valuable mix of materials that fits within a knapsack of fixed capacity. When the value functions of the involved materials are fully known and differentiable, the most valuable mixture can be found by direct application of Lagrange multipliers.

In this paper we introduced Gaussian Process based Optimistic Knapsack Sampling (GPOKS) — a novel model-based reinforcement learning scheme for solving stochastic fractional knapsack problems. The scheme is founded on Gaussian Process (GP) enabled Optimistic Thompson Sampling (OTS). Our empirical results demonstrates that this scheme converge significantly faster than LAKG. Furthermore, GPOKS incorporates GP based learning of the material unit values themselves, forming the basis for OTS supported balancing between exploration and exploitation. Using resource allocation in web polling as a proof-of-concept application, our empirical results show that GPOKS consistently outperforms LAKG, the current top-performer, under a wide variety of parameter settings.

In our further work, we will address games of interacting GPOKS for solving networked and hierarchical resource allocation problems. Furthermore, we are investigating techniques for decomposing the GP calculations for increased computational performance.

References

Black, P. E. 2004. Fractional knapsack problem. *Dictionary of Algorithms and Data Structures*.

Bretthauer, K. M., and Shetty, B. 2002. The Nonlinear Knapsack Problem — Algorithms and Applications. *European Journal of Operational Research* 138:459–472.

Granmo, O.-C., and Berg, S. 2010. Solving Non-Stationary Bandit Problems by Random Sampling from Sibling Kalman Filters. In *Proceedings of the Twenty Third International Conference on Industrial, Engineering, and Other Applications of Applied Intelligent Systems (IEA-AIE 2010)*, 199–208. Springer.

Granmo, O.-C., and Oommen, B. J. 2006. On Allocating Limited Sampling Resources Using a Learning Automata-based Solution to the Fractional Knapsack Problem. In *Proceedings of the 2006 International Intelligent Information Processing and Web Mining Conference (IIS:IIPW'06)*, Advances in Soft Computing. Springer.

Granmo, O.-C., and Oommen, B. J. 2010. Solving Stochastic Nonlinear Resource Allocation Problems Using a Hierarchy of Twofold Resource Allocation Automata. *IEEE Transactions on Computers* 59(4):545–560.

Granmo, O.-C.; Oommen, B. J.; Myrer, S. A.; and Olsen, M. G. 2006. Determining Optimal Polling Frequency Using a Learning Automata-based Solution to the Fractional Knapsack Problem. In *Proceedings of the 2006 IEEE International Conferences on Cybernetics & Intelligent Systems (CIS) and Robotics, Automation & Mechatronics (RAM)*. IEEE.

Granmo, O.-C.; Oommen, B. J.; Myrer, S. A.; and Olsen, M. G. 2007. Learning Automata-based Solutions to the Nonlinear Fractional Knapsack Problem with Applications to Optimal Resource Allocation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 37(1):166–175.

Granmo, O.-C. 2010. Solving Two-Armed Bernoulli Bandit Problems Using a Bayesian Learning Automaton. *International Journal of Intelligent Computing and Cybernetics* 3(2):207–234.

Gupta, N.; Granmo, O.-C.; and Agrawala, A. 2011a. Successive Reduction of Arms in Multi-Armed Bandits. In *Proceedings of the Thirty-first SGAI International Conference on Artificial Intelligence (SGAI 2011)*. Springer.

Gupta, N.; Granmo, O.-C.; and Agrawala, A. 2011b. Thompson Sampling for Dynamic Multi-Armed Bandits. In *Proceedings of the Tenth International Conference on Machine Learning and Applications (ICMLA'11)*. IEEE.

Kellerer, H.; Pferschy, U.; and Pisinger, D. 2004. *Knapsack Problems*. Springer.

May, B. C.; Korda, N.; Lee, A.; and Leslie, D. S. 2012. Optimistic bayesian sampling in contextual-bandit problems. *J. Mach. Learn. Res.* 8:2069–2106.

Narendra, K. S., and Thathachar, M. A. L. 1989. *Learning Automata: An Introduction*. Prentice Hall.

Pandey, S.; Ramamritham, K.; and Chakrabarti, S. 2003. Monitoring the Dynamic Web to Respond to Continuous Queries. In *12th International World Wide Web Conference*, 659–668. ACM Press.

Rasmussen, C. E., and Williams, C. K. I. 2006. *Gaussian Processes for Machine Learning*. The MIT Press.

Srinivas N., Krause A., K. S., and M., S. 2010. Gaussian process optimization in the bandit setting: No regret and experimental design. In Fürnkranz, J., and Joachims, T., eds., *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 1015–1022. Haifa, Israel: Omnipress.

Thompson, W. R. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika* 25:285–294.

Wolf, J. L.; Squillante, M. S.; Sethuraman, J.; and Ozsen, K. 2002. Optimal Crawling Strategies for Web Search Engines. In *11th International World Wide Web Conference*, 136–147. ACM Press.

Wyatt, J. 1997. *Exploration and Inference in Learning from Reinforcement*. Ph.D. Dissertation, University of Edinburgh.

Scheme	p_1/p_2	Avg[#Updates] t=10	Avg[#Updates] t=100	Avg[#Updates] t=1000
Optimal	0.90/0.10	9.1	91.0	909.9
Uniform	0.90/0.10	5.9	59.0	590.0
LAKG	0.90/0.10	6.0	71.6	874.9
GPOKS	0.90/0.10	8.0	88.9	903.0
GPOKS-Mean	0.90/0.10	8.5	89.7	902.9
Optimal	0.75/0.25	8.1	81.2	812.5
Uniform	0.75/0.25	6.9	68.8	687.5
LAKG	0.75/0.25	6.9	74.1	793.1
GPOKS	0.75/0.25	7.4	78.8	807.9
GPOKS-Mean	0.75/0.25	6.6	69.6	792.2
Optimal	0.55/0.45	7.5	75.2	752.5
Uniform	0.55/0.45	7.5	74.8	747.5
LAKG	0.55/0.45	7.5	74.8	749.8
GPOKS	0.55/0.45	7.0	73.5	749.4
GPOKS-Mean	0.55/0.45	5.4	52.8	725.3

Table 1: Average number of updates at different times, $w_\sigma = 0.1$

Scheme	p_1/p_2	$w_\sigma = 0.0$	$w_\sigma = 0.2$	$w_\sigma = 0.4$
GPOKS	0.75/0.25	808.2	804.5	804.1
GPOKS-Mean	0.75/0.25	793.9	787.2	769.1

Table 2: The performance of GPOKS variants under different levels of white noise