Radboud University Nijmegen

A Bachelor Thesis in
Artificial Intelligence

Optimizing the exploration-exploitation trade-off of Lock-in Feedback

Supervisor:
Maurits Kaptein
Donders Institute for
Brain, Cognition and
Behaviour

Author:
Moira Berens
s4221826

Second Reader:
Louis Vuurpijl
Donders Institute for
Brain, Cognition and
Behaviour

April 4, 2016

Abstract

Over the years several strategies to solve bandit problems have been discovered and examined. Strategies that deal with continuum-armed bandit problems, which is a variant of a bandit problem, are less frequently researched. In this study Lock-in Feedback (LiF), a strategy that deals with continuum-armed bandit problems is optimized. The main advantage of LiF over other continuum-armed bandit strategies is the capability to deal with concept drift. However, the oscillation needed to detect the concept drift makes LiF less efficient. The aim of this study is to adapt LiF in such a way, that LiF is still able to detect concept drift, with using less oscillations. So the main research questions that will be answered is: Could we adapt the policy of LiF such that it needs fewer oscillations to detect concept drift in order to reduce its linear regret? Two simulation studies are done to answer this research question. In the second study different "stabilization policies" were tested. The aim of the stabilization polices was to detect concept drift with the use of less oscillations. The results of this study show that the stabilization policies are both able to detect concept drift, but more research should be done to increase the accuracy of these stabilization policies.

Introduction

If you want to buy a new cell phone, it is important to keep an eye on the available offers. When an offer appears that seems worthwhile, acting quickly is important. Just thinking about the deal for a couple of hours, could result in a worse deal. Online deals are changing every minute depending on competition, custom profile, and supply and demand. This phenomenon is also called dynamic pricing and is the result of an important purpose of a company; profit. To reach this aim, the "best" price of a product should be found at every time point, where best can be quantified as the price that gives the highest profit. Finding this price is a difficult task.

To solve this problem we could try to model the problem as a function of multiple variables, where the variables represent the possible dependencies and the output of that function would be the best price of a certain product. Theoretically no function could take into account all the possible dependencies. Moreover, most of these dependencies are unknown in practice.

The problem described above is also known as a continuous maximization problem. These problems do not only appear in the retail sector, but in a wider range of sectors. Practical settings where this problem also arises include controlling the temperature of a chemical reaction that maximizes the yield and maximizing the amount of bits transmitted per minute over a noisy channel.[18] The latter problem is represented in Figure 1.

In this representation x stands for the amount of bits transmitted per minute and y stands for the amount of bits received per minute. The aim is to maximize the amount of received bits. Increasing x does not always lead to an increasing of y. Namely, if the x is too high the bits cannot be differentiated from the noise anymore. Despite the fact that continuous maximization problems appear often there is no single agreed method up to now to solve maximization problems.

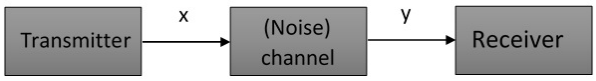

Figure 1: Simplified representation of noisy channel maximization problem

In the remainder of this paper the dynamic pricing problem is used as a guideline. A simplified representation of the dynamic pricing example is given in Figure 2. Figure 2 shows a plot of the profit against the price of a certain product at a certain time point. In this plot the best price to ask is 150 euro, then the profit would be around 100 euro. The aim of the company is to find that best price.

A continuous maximization problem can be formalized as $\vec{x}_{max} = argmax_{\vec{x}} f(\vec{x})$. Where $f(x)$ is a function which is unknown and x is a controllable vari-

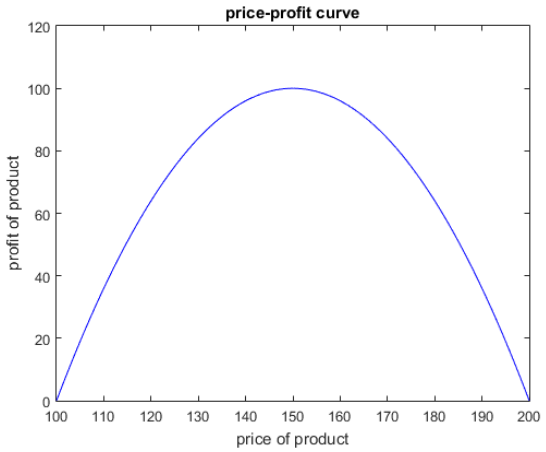

Figure 2: Simplified representation of a price-profit curve

able. The aim is to find the instance of x that maximizes $f(x)$. In the remainder of this paper the x-value which maximizes $f(x)$ will be called x_{max}. Continuous maximization problems could be solved by sequential experiments. These experiments consist of a sequence of sub-experiments. Where the outcome of the previous sub-experiment influences the next sub-experiment. During such an experiment x_{max} can be found by testing different x-values.

If we are not only interested in finding x_{max}, but also in the efficiency of the search then a continuous maximization problem could be framed as a bandit problem. Bandit problems are problems where at each time point several possible actions are to experiment, for example several x-values. The name bandit comes from the fact that these problems are modeled as multi-armed bandits, which is a slot machine with multiple arms. Each of the arms have a different pay-off probability distribution. Before the experiment these distribution are unknown. By a sequential experiment the distributions should be discovered. The aim is not only to find by sequential experiment the best arm, and thus x_{max}, but also in the most efficient way. Efficiency can be reached by optimizing the exploration-exploitation trade-off. This crucial trade-off between exploration, getting more information about the pay-off distributions of the arms, and exploitation, choosing the expected x_{max}, is one of the biggest problems faced in bandit problems.[17, 19]

In the dynamic pricing problem described above, the aim is not only to find x_{max}, but also to find it in the most efficient way. Efficiency is quantified in terms of regret during this study. The equation of regret is given in Equation 1, where τ is the amount of sequential experiments.

$$R = \sum_{t=1}^{\tau} f(x_{max}) - f(x_t). \tag{1}$$

The option for this quantification of efficiency is made, since it clearly illustrates the total profit the company has lost during the sequential experiment due to exploration. It is important to find x_{max} in an efficient way, since less efficiency leads to less profit. Therefore the dynamic pricing problem will be framed as a bandit problem in this study. If we frame the dynamic pricing problem as bandit problem, the arms of the bandit represent the different prices and the pay-off probability distributions of the arms are represented by the observed profit.

There exist several kinds of bandit problems. The multi-armed bandit problem and the continuum-armed bandit problem proposed by Rajeev Agrawal are the most famous ones.[1]

Framing the dynamic pricing problem as a multi-armed bandit problem has two disadvantages compared to framing it as a continuum-armed bandit problem.

Firstly, if we want to frame the dynamic pricing problem as a multi-armed bandit problem, there should be an arm for all the possible prices, $prices \in \{0, \ldots, \infty\}$. This would lead to an infinite amount of arms, which should at least be tested once during an experiment to be able to make a trustworthy guess which arm would have the highest pay-off distribution. To test all the different arms the experiment should run infinitely. In practice this will never happen, due to the time-limit set to the experiment, which is also known as the horizon. A solution to this infinite arm problem is dividing the range of

the controllable variable in equal lengths and play a multi-armed bandit problem over the discretized problem.[8] A consequence of this solution is that there is a chance that the best instance of the controllable variable cannot be discovered, if the lenght of the subintervals is too large.

Secondly, the loss of information is an important reason not to opt for framing the dynamic pricing problem as a multi-armed bandit problem. Since one of the characteristics of a multi-armed bandit problem is that the pay-off distribution of the arms are independent of each other.[4] In the dynamic pricing problem this is not the case, prices that are close to each other yield approximately similar expected pay-off distributions. This information is unused if we frame the dynamic pricing problem as a multi-armed bandit problem.

Framing the dynamic pricing problem as a continuum-armed bandit problem would have two advantages. Firstly, an infinite amount of arms can be used. Secondly, this context does not make the assumption that arms are independent of each other. In this study the dynamic pricing problem is framed as a continuum-armed bandit problem.

In this study the Lock in Feedback strategy (LiF), designed by M. Kaptein and D. Iannuzzi [6], is used to find x_{max}. LiF tries to find x_{max} by sequential experiment. This strategy starts by examining the slope of a certain x-value (x_c). If the slope of that certain x_c is smaller than 0, x_c will decrease. If the slope is higher than 0, x_c will increase. The amount of increasing or decreasing is dependent on the approximated slope. To determine the slope at a certain x_c LiF makes use of the lock-in amplifier technique, which is widely used in other fields.[11] This technique lets the controllable variable oscillate as a cosine function, which is shown in Figure 3. In the figure the center of the oscillation, x_c, is 120 euro and the amplitude of the oscillation is 5. In order to determine the slope at the price x_c LiF makes a cosine oscillation around x_c. This oscillation is divided in a predetermined amount of evenly distributed points, the x-values of these points will be called Δx_c. The y-values of these Δx_c-values are observed and multiplied by the distance between x_c and the Δx_c. Averaging these results gives the slope at x_c. Even with large noise on the data stream LiF is able to approximate the slope.[6]

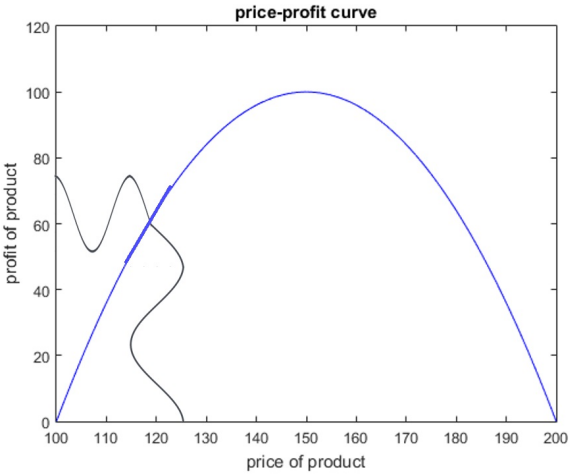

Figure 3: Simplified representation of a oscillation of the controllable variable, the price of a product

The choice for this strategy is made because it is able to deal with concept drift.[3] Concept drift means that the function changes over time, Figure 4 shows a simplified price-profit curve that suffers from concept drift. This phenomenon also happens in the real world due to the price being dependent on external factors. Just due to weather or competition prices can change over time. Therefore it is very important that a strategy is able to deal with concept drift. Another advantage of LiF is that it is able to deal with situations where noise levels are high.[6]

During the project of M. Kaptein and D. Iannuzzi[6] LiF was compared with other well-known strategies that are used to solve continuum-armed bandit problems. Their results show that LiF was very efficient in finding x_{max}. But in the long run LiF has linear regret due to oscillations, because LiF remains oscillating even if the x_{max} has already been found. So if we want to optimize LiF we should try to reduce oscillations after x_{max} has been found. Just stop with oscillation when LiF has found x_{max} is not an ideal solution to this problem. The oscillations

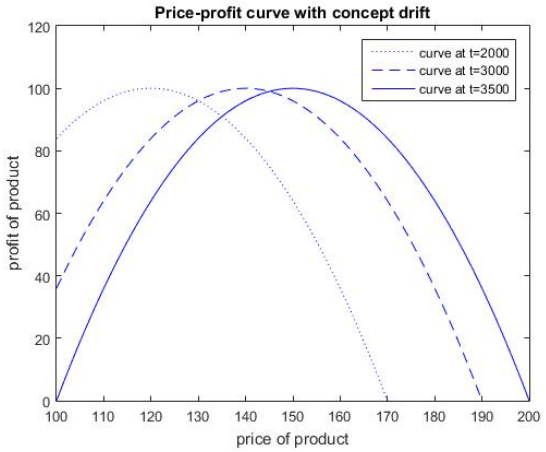

Figure 4: Simplified representation of a price/profit curve with concept drift

are not only used by LiF during the search to x_{max} but also afterwards to detect concept drift. So if we want to improve LiF we should try to find another solution. The question that will be answered in this study is: Could we adapt the policy of LiF such that it needs less oscillations to detect concept drift in order to reduce its linear regret?

In the following section of this paper more strategies for continuum-armed bandit problems and their advantages and disadvantages are described. Then both the method and the simulator used during the experiment are explained in more detail. Subsequently, the results of the experiments will be described and discussed. Given the obtained results during this study LiF will be modified and compared with other well-known continuum-armed bandit strategies. Finally the research question will be answered and the opportunities of LiF will be discussed.

Literature review

Over the years several strategies to solve the multi-armed bandit problem are discovered and examined. Examples are probability matching strategies, UCB, the Poker strategy, and semi-uniform strategies as ϵ-greedy.[12, 14] Strategies that solve continuum-armed bandit problems are less examined and a solution for continuum-armed bandit problems seems to be much harder to find. While in multi-armed bandit problems logarithmic regret is achievable, in continuum-armed bandit problems polynomial regret is the lower limit.[13] Recently, strategies to (approximately) solve continuum-armed bandit problem have been studied by a number of authors.[7, 8, 13]

One Strategy that is examined by different researchers is Thompson Sampling.[5, 19, 7, 9] Thompson sampling is a probability matching strategy originally made to approximately solve multi-armed bandit problems. This strategy tries to determine the prior probability distribution of every arm taking into account the available (prior) information. The probability distribution is updated after every sequential experiment.[2] However, Thompson Sampling was originally used as a multi-armed bandit strategy. This strategy could be easily transformed to an strategy that was able to deal with continuum-armed bandit problems.[5]. In Algorithm 1 an continuum-armed variant of Thompson Sampling is given. This variant starts with sampling for a predefined amount of time, which is given to Thompson sampling as input variable *samples*. After this sampling period a two-degree polynomial is fitted thought the obtained data points. This two-degree polynomial is the best fitted polynomial in a sense of least-squares given the samples. Given this two-degree polynomial and the samples the standard error is calculated for all the coefficients of the polynomial.

The obtained coefficients and their standard errors are used to make a normal-distribution, where the means are equal to the coefficients and the standard deviations are equal to standard errors. Note that the standard error will decrease with an increasing of the amount of samples, assuming that no concept drift happened.

Then three normal-distributions will be made, one for every β-value, and from all of these distributions a sample will be drawn. These obtained values will be used to make a new 2-degree polynomial.

The extremum of this created 2-degree polynomial will be calculated and the x-value belonging to this

extremum is used as the next x-value for the sample. Given this sample a new 2-degree polynomial will be created and this does go on and on until the end of the horizon. Note that this variant of Thompson Sampling makes the assumption that the continuum-armed bandit problem could be approximated by a 2-degree polynomial.

Algorithm 1 Thompson Sampling

Require: samples, xMin, xMax, τ

1: **for** t = 1, ..., samples **do**
2: draw $\sim \mathcal{U}(0, 1)$
3: $x.add((xMax - xMin) \times draw + xMin)$
4: $y.add(f(x.getLastItem()) + \epsilon_t)$
5: **end for**
6: **for** t = samples, ..., τ **do**
7: $[\beta, SE] \leftarrow$ fit 2-degree polynomial through data samples obtained until t {Where β is a array with three values representing the coefficient of polynomial and SE the standard error of the three coefficients}
8: $\beta 1 \sim \mathcal{N}(\beta[1], SE[1])$
9: $\beta 2 \sim \mathcal{N}(\beta[2], SE[2])$
10: $\beta 3 \sim \mathcal{N}(\beta[3], SE[3])$
11: extremum $\leftarrow$ calculate x-coordinate of extremum of the polynomial with coefficients $\beta 1, \beta 2, \beta 3$
12: $x.add(extremum)$
13: $y.add(f(extremum) + \epsilon_t)$
14: **end for**

Thompson sampling has not been popular in the literature until recently. Researchers found out that Thompson sampling is highly effective for balancing exploration and exploitation.[16] Since Thompson sampling is able to make a balance between exploitation and exploration, this strategy is able to find x_{max} in a very efficient way.

However, Thompson Sampling can be seen as one of the most efficient strategy to deal with continuum-armed bandit problems, this strategy needs more knowledge about the continuum-armed problem than other strategies. Both the model and the x-range of the curve are required as a input argument or should be assumed. In case of dynamic pricing, you could imagine that approximating the x-range is possible. But approximating the model, by approximating the degree of the polynomial that fits the price-profit curve the best in terms of the least-squares is impossible in practice. A wrong assumption would make Thompson Sampling less efficient after all.

Besides the fact that Thompson Sampling needs a lot of information about the curve, Thompson sampling is also not able to deal with situations where the curve suffers from concept drift. The reason why Thompson sampling is quite bad in detecting the concept drift has to do with the fact that Thompson sampling uses all the samples obtained during the whole run to fit the next polynomial. Assume that concept drift did happen and that x_{max} shifted from 150 tot 190. Thompson Sampling will use both the samples obtained before the concept drift and after the concept drift to fit the 2-degree polynomial. So Thompson Sampling will approximate x_{max} somewhere around 170, which is not a approximation close the x_{max} at all. Some researchers have already been trying to deal with this disadvantage of Thompson Sampling.[15]

Other strategies that were originally made for multi-armed bandit problems, but are also usable to solve continuum-armed bandit problems are the semi-uniform strategies. This group of strategies is one of the oldest and simplest strategies that were able to approximately solve bandit problems. These strategies make a clear distinction between exploration and exploitation. Exploitation is mostly defined as taking the best possible action known so far, whereas exploitation is usually taking actions with uniform probability.[10]

ϵ-first belongs to the class of semi-uniform strategies and is one of the simplest of these class. ϵ-first starts with a pure exploration phase, which is followed by an exploitation phase. The length of the exploration phase is ϵN and the length of the exploitation phase is (1- ϵ)N. Where N is the horizon of the strategy, which means that N times an x-value is selected. ϵ is a value which is typically 0.1.[10]

Although, ϵ-first strategy was originally made for multi-armed bandit problems only two small adaptions transformed this strategy to a continuum-armed

bandit strategy. The first adaption was that the strategy has to choose a random x-value instead of an arm. Secondly, before the exploitation phase could start a method should fit a polynomial through the set of data points obtained during the exploration phase. The polynomial is fit through the data points in terms of least-squares error. On the basis of this polynomial the expected x_{max} should be determined. The disadvantage of ϵ-fist is that it is not able to deal with concept drift and that ϵ-first needs, just as Thompson Sampling, both the model and the x-range of the curve. In Algorithm 2 the pseudo-code of the continuum-armed bandit variant of ϵ-first is given. Note that this strategy makes the assumption that the continuum-armed bandit problem is approachable by a 2-degree polynomial.

Algorithm 2 ϵ-first

Require: $\epsilon, xMin, xMax, \tau$

1: **for** $t = 1, ..., \epsilon \times \tau$ **do**
2: draw $\sim \mathcal{U}(0, 1)$
3: $x.add((xMax - xMin) \times draw + xMin)$
4: $y.add(f(x.getLastItem()) + \epsilon_t)$
5: **end for**
6: $[\beta 1, \beta 2, \beta 3] \leftarrow$ fit 2-degree polynomial through data samples obtained {Where the three β-values represent the coefficient of polynomial}
7: extremum $\leftarrow$ calculate x-coordinate the extremum of the polynomial with coefficients $\beta 1, \beta 2, \beta 3$
8: **for** $t = \epsilon \times \tau, ..., \tau$ **do**
9: $f(extremum) + \epsilon_t$
10: **end for**

LiF, in contrast to the above described strategies, is especially made for continuum-armed bandit problems. There exists two variants of this strategy, both these variants will be explained shortly.

Both variants of LiF have the following tuning parameters: x_c the center of the first oscillation, A the amplitude (this is the amplitude of the oscillation), γ the learning rate and T the integration time. The integration time stands for the amount of evenly distributed points that are chosen on the oscillation. Note that the model and the x-range are not required as input variables.

The difference between the variants is the way they update, LiF-I updates after every oscillation and thus uses a batch approach, while LiF-II updates after every observation and thus makes use of on-line learning. The pseudo-code for both strategies are described in Algorithm 3 and Algorithm 4. The only difference between these strategies is line 6 and the initialisation of y_ω^Σ or $\vec{y}_\omega$.[6] During this study only LiF-II will be used, the reason for this decision will be explained in the method.

Algorithm 3 LiF-I using batch learning

Require: $x_c, A, T, \gamma, y_\omega^\Sigma = 0, \tau$

1: $\omega = \frac{2\pi}{T}$
2: **for** $t = 1, ..., \tau$ **do**
3: $x_t = x_c + A\cos\omega t$
4: $y_t = f(x_c + A\cos\omega t) + \epsilon_t$
5: $y_\omega^\Sigma = y_\omega^\Sigma + y_t \cos\omega t$
6: **if** $(t \bmod T == 0)$ **then**
7: $y_\omega^* = y_\omega^\Sigma / T$
8: $x_c = x_c + \gamma y_\omega^*$
9: $y_\omega^\Sigma = 0$
10: **end if**
11: **end for**

Algorithm 4 LiF-II using online learning

Require: $x_c, A, T, \gamma, \vec{y}_\omega = \{NA_1, ..., NA_T\}, \tau$

1: $\omega = \frac{2\pi}{T}$
2: **for** $t = 1, ..., \tau$ **do**
3: $x_t = x_c + A\cos\omega t$
4: $y_t = f(x_c + A\cos\omega t) + \epsilon_t$
5: $\vec{y}_\omega =$ push $(\vec{y}_\omega, y_t \cos\omega t)$
6: **if** $(t > T)$ **then**
7: $y_\omega^* = (\Sigma \vec{y}_\omega)/T$
8: $x_c = x_c + \gamma y_\omega^*$
9: **end if**
10: **end for**

The advantages of LiF are already mentioned above. Namely, LiF is able to deal with concept drift, does not need the model or the x-range. However LiF, does also have some disadvantages.

Firstly, several variables should be initialized. Badly initialized variables could lead to slower convergence or no convergence at all. For example, a badly initialized learning rate could lead to no convergence at all. This could happen if the learning-rate is too large and LiF "overshoots" x_{max}. In case the difference between x_{max} and the previous x_c is smaller than the difference between x_{max} and the update x_c, LiF has overshoot x_{max}. So it is important to have a small learning-rate, but smaller learning-rates lead to slower convergence and thus to more regret.

Moreover if the amplitude is very small, LiF could get stuck in a local maximum or at a flatter part of the curve. However, a larger amplitude will lead to higher linear regret after that LiF is stabilized around x_{max}.

Finally, the increasing of the integration time leads to smoother updates, but slower convergence. So a too large integration time could make LiF less efficient.

So to make LiF optimal, these variables should be optimized. The optimization of these variables is depending on the curve. In practice the curve is unknown and optimization based on the curve is not possible. So to make LiF the most efficient in practice, the most robust settings should be used. So a relative large amplitude and a small learning would be the most robust.

In the study of Kaptein and Iannuzzi[6] LiF was compared both with Thompson Sampling and ϵ-first. They used both Thompson sampling and ϵ-first, since these strategies put LiF in perspective. As already mentioned Thompson sampling is very efficient in the search and can be seen as a lower-regret bound for continuum-armed bandit problems.[5] In contrast to ϵ-first, which can be seen as the upper regret bound for continuum-armed bandit problems. Kaptein and Iannuzzie concluded that LiF is very efficient in the search. But from the time point that x_c is (almost) equal to x_{max}, LiF suffers from linear regret. This linear regret is due to the oscillation.

Method

In this section the simulator and methods used during this study are described. Firstly, the overall approach and the simulator will be explained in more detail. Subsequently, the adaptations that will be made to LiF are described and how these adaptations will be tested using the simulator. LiF with the best top-detection method and the stabilization policy will be called the *modified LiF*, where best is quantified in terms of regret. This modified LiF will be compared with both Thompson Sampling and ϵ-first.

Given the results obtained during the study of Kaptein and Ianuzzie[6] the policy of LiF should be changed after that x_c approximates x_{max}. To be able to do this LiF should be able to detect this situation where x_c approximates x_{max}. If x_c approximates x_{max}, x_c actually approximates the extremum of continuous maximization problem. Therefore we will call the method added to LiF, which is responsible for detecting if x_c approximates x_{max} a *Top-detection method*. During this study two different top-detection methods will be examined. Both these methods will be explained in more detail below. Both these top-detection methods only uses information obtained by the oscillations of LiF to decide if x_c approximates x_{max}, thus no further information has to be obtained and this keeps the regret as low as possible.

After that LiF has detected that x_c is (almost) equal to x_{max}, LiF will call a *stabilization policy*. This stabilization policy will take over the control over the sub-experiments. The aim of the stabilization policy is to detect concept drift with less oscillations than LiF. So the stabilization policy tries to exploit as much as possible, while keep track of concept drift. If the stabilization policy detects concept drift, the stabilization policy will *notify* LiF by giving back the current time point. LiF will take the control over the sub-experiment again and will search for x_{max}. This will go on and on until the end of the horizon. During this study two different stabilization policies will be tested. Both these stabilization policies will be explained in more detail below, after the simulator and the top-detection methods have been described.

Simulator

During this study a simulator is used tot test the different top-detection methods and the stabilization policies. In this section the simulator will be described in more detail.

A 2-degree polynomial, which represent a simplified price-profit curve, is used in this study as the continuum-armed bandit problem. The function of the polynomial is:

$$f(x) = -0.04x^2 + 12x + 800 + \epsilon, \epsilon \in \mathcal{N}(0, 5). \quad (2)$$

This price-profit curve suffers from concept drift during the run, so Equation 2 is only used if no concept drift has happened. Concept drift was a requirement of the simulator, since the modified LiF should be able to deal with concept drift. There were several drifts during the run:

- between t = 2000 and t = 2400, curve shifts 10 to left
- between t = 3400 and t = 4200, curve shifts 20 to right
- between t = 5200 and t = 6400, curve shifts 30 to left
- between t = 7400 and t = 9000, curve shifts 40 to right

As you can see the speed did not differ between the drifts, but the amount of the shifted positions did. During this study it has been decided not to change the speed of the drifts, because the interest is not in how the stabilization policies deal with the speed variations of the drifts. But during this study we are interested if the stabilization policy is able to detect shifts even if they are very small.

Between every shift there where 1000 time points with no drifts, this was done such that there was enough time for both LiF to search for the new x_{max} and for the stabilization policy to take over the control of the sub-experiments. If the intervals between the drifts where too small, there would be no time for the stabilization policy to control the sub-experiments and we would not have been able to draw any conclusion on how they function. Figure 5 shows the different curve positions after the drifts.

A duration of a whole run was 10.000 time points, in the remainder of this paper this will be called the horizon. Every setting was tested 50 times, this was done to minimize the influence of the noise on the results.

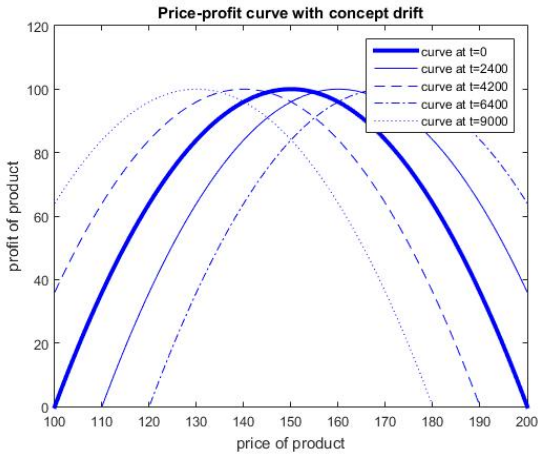

Figure 5: The curve used during this study at different time points

During this study only LiF-II, the one with the online learning approach, will be optimized. We decided for this variant, because this variant is more robust to erroneous selection of the amplitude than LiF-I.[6] Especially, when the amplitude is large LiF-I has the tendency to become unstable if x_c approximates x_{max}.[6] To make LiF more robust it is important to have a large amplitude during the search, this gives LiF the possibility to overcome small local maximum or flatter parts in the curve. So during the search a large amplitude is to the advantage of a smaller amplitude. However, stability in the top is also important because the top-detection method should be able to detect if x_c approximates x_{max}. Therefore, LiF-II is best option to choose here. In the remainder of this paper if stated LiF, LiF-II is

meant. In this study LiF has the following parameters:

- $x_c = 110$
- Amplitude = 10
- Learning-rate = 0.1
- Integration time = 20

As already mentioned during this study LiF will be compared with Thompson Sampling and ϵ-first. The pseudo-code of the strategies has already been given in Algorithm 1 and 2. Both these strategies make the assumption that the price-profit curve could be approximated with a 2-degree polynomial, which is exactly the case. So we could say that both Thompson Sampling and ϵ-first has an unfair advantage in comparison with LiF, since LiF does not have this information. The parameters for Thompson sampling used during this study are:

- samples = 100
- xMin = 100
- xMax = 200

The parameters of ϵ-first are:

- $\epsilon = 0.1$
- xMin = 100
- xMax = 200

In Figure 7 LiF is compared with both Thompson Sampling and ϵ-first. The simulator was used to generate this Figure, but the curve did not suffer from concept drift during the run. So the polynomial used during the whole run was equal to Equation 2. There was no concept drift on the curve, since both Thompson Sampling and ϵ-first are not able to deal with concept drift. The results in Figure 7 agree with the results obtained during the research of Kaptein and Iannuzzi.[6] So during the search LiF is very efficient, which can be seen in the left upper corner. But after stabilization LiF suffers from linear regret. The cumulative regret at the end of the horizon is equal to 2.3×10^4. So the aim of this study is to improve this regret, so reduce it by adding a top-detection method and a stabilization policy.

Top-detection method 1: mean-dependent

The mean-dependent top-detection method (TD1) uses the observed y-values during the last oscillation to determine if x_c is close to x_{max}. TD1 divides the observed y-values in three different groups: The first group contains the y-values observed at x_c, or if the integration time is not divisible by four the y-values observed at the x-value(s) closest to x_c. The remainder of the y-values will be divided in two different groups, one group with the y-values observed at x-values left to x_c and in the other group the y-values observed at x-values right to x_c. In Figure 6 these groups are illustrated in a simplified manner, by dotted lines. The groups are provided with a group number to make the further explanation clearer.

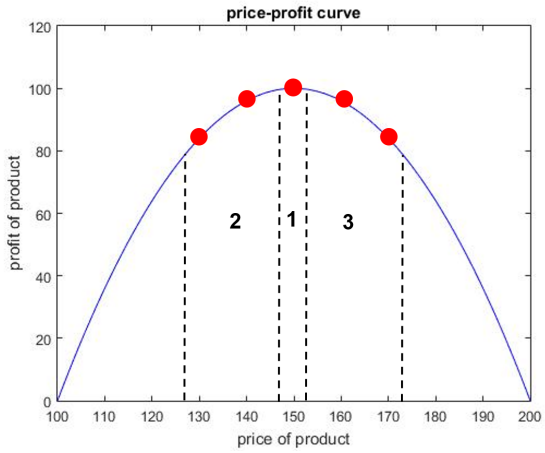

Figure 6: Simplified representation of the three groups made by the TD1. The red dots represent the observed y-values and the numbers represent the group numbers

The mean of the observed y-values of all these groups will be calculated. If the observed y-value for both groups 2 and 3 is lower than the mean of the observed y-values of group 1, the top is detected. Due to noise it could happen that a top is detected at a certain x_c, whereas x_c is not even close to x_{max}. This could happen if the average noise in the groups 2 and

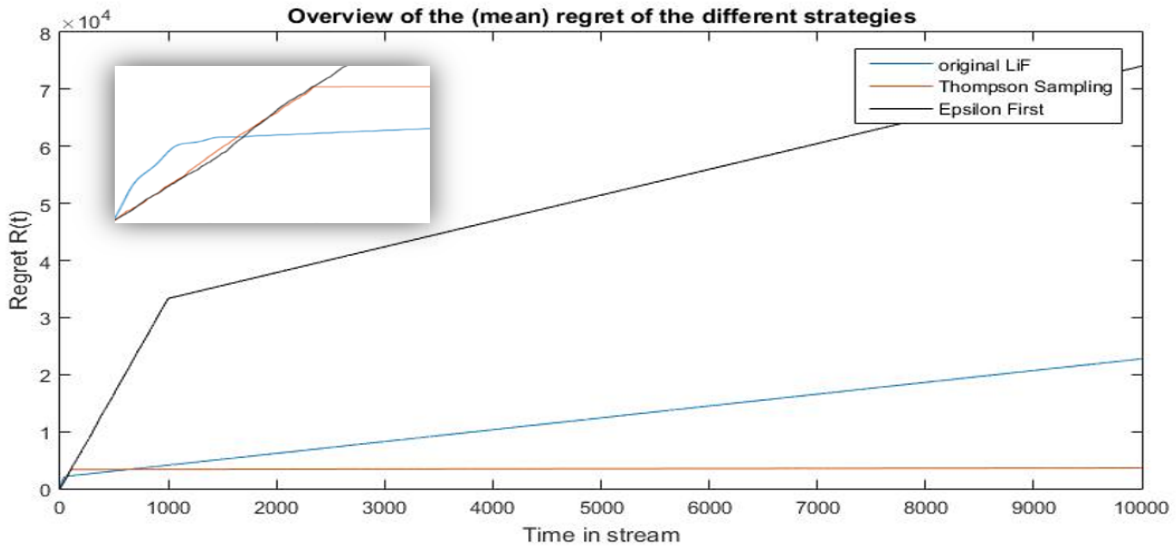

Figure 7: LiF compared with Thompson Sampling and ϵ-first

3 is negative ($\epsilon < 0$) and the average noise in group 1 is positive ($\epsilon > 0$). To prevent TD1 from detecting the top at a wrong position. The top should be detected several times in a row, before the top-detection is detected definitely. This predefined variable will be optimized, so $row \in \{5, 10, 15, 20, 15, 30\}$.

The width of the groups 2 and 3 are dependent of the amplitude of LiF. So if the amplitude of LiF is smaller the width of groups 2 and 3 is also smaller. If the amplitude is too small TD1 is not able to make a detection at all, since the group averages are to close to each other, which is a disadvantage of TD1. The pseudo-code of TD1 is given in Algorithm 6, in Algorithm 5 you can see how TD1 is called by LiF.

Top-detection method 2: slope-dependent

The slope-dependent top-detection method (TD2) makes use of the approximated slope by LiF to determine whether x_c is close to x_{max}. This approximated slope is proportional to the real slope. The variable y^*_ω approximates the slope of x_c. If y^*_ω slope is smaller than a predefined value, this variable will be called

Algorithm 5 LiF in combination with TD1

Require: x_c, A, T, γ, $\vec{y}_\omega = \{NA_1, ..., NA_T\}$, rowList, row, slope, start = 0, τ

$\omega = \frac{2\pi}{T}$
for t = 0, ..., T **do**
  $cosValue(t) = cos(w * t)$
end for
for $t = 1, ..., \tau$ **do**
  $x_t = x_c + A\cos\omega t$
  $y_t = f(x_c + A\cos\omega t) + \epsilon_t$
  $\vec{y}_\omega$ = push $(\vec{y}_\omega, y_t \cos\omega t)$
  $\vec{y}_t(t \bmod T) = y_t$
  if $(t > T + start)$ **then**
    $y^*_\omega = \Sigma\vec{y}_\omega / T$
    $rowList(t \bmod row) = TD1(\vec{y}_t, cosValue)$
    if $\Sigma rowList == row$ **then**
      [t] = perfectSP(t, x_c, τ)
      $start = t$
      set all values in rowList to zero
    else
      $x_c = x_c + \gamma y^*_\omega$
    end if
  end if
end for

Algorithm 6 TD1

Function TD1 ($\vec{y}_t, cosValue$)
meanMiddle ← Calculate mean of values in $\vec{y}_t$ at indices included in middle
left ← Find indices of items in cosValue smaller than zero
meanLeft ← Caclulate mean of values in $\vec{y}_t$ at indices included in left
right ← Find indices of items in cosValue larger than zero
meanRight ← Caclulate mean of values in $\vec{y}_t$ at indices included in right
return if meanLeft < meanMiddle > meanRight else 0

slope, the top is detected. Several *slope* values will be tested. $slope \in \{0.1, 0.3, 0.5, 0.7\}$

It could happen that the approximated slope is smaller than the predefined *slope* when x_c is not x_{max} or close to x_{max}. This could happen due to noise or to large learning-rates. In case of noise, it happens for example if the real slope of x_c is ascending and the averaged noise left to x_c is positive ($\epsilon > 0$), whereas the averaged noise right to x_c is negative, ($\epsilon < 0$). In this case the obtained y-values during the oscillation are close to each other and the slope will be quite small and it looks like the x_c approximates x_{max}.

Secondly, if the large learning-rate it too large, it could happen that the update of x_c is too large, then a certain x-value is observed several times during one oscillation. This will also lead to a small amplitude.

To prevent the TD2 from making wrong detections, it is important that the top is detected several times in a row. This variable row will be optimized too. $row \in \{5, 10, 15, 20, 15, 30\}$. In Algorithm 7 the psuedo-code of TD2 is given.

Algorithm 7 LiF in cobination with TD2

Require: x_c, A, T, γ, $\vec{y}_\omega = \{NA_1, ..., NA_T\}$, rowList, row, slope, start = 0, τ
$\omega = \frac{2\pi}{T}$
for $t = 1, ..., \tau$ **do**
  $x_t = x_c + A\cos\omega t$
  $y_t = f(x_c + A\cos\omega t) + \epsilon_t$
  $\vec{y}_\omega$ = push $(\vec{y}_\omega, y_t \cos\omega t)$
  if $(t > T + start)$ **then**
    $y^*_\omega = \Sigma\vec{y}_\omega / T$
    $rowList(t \bmod row) = |y^*_\omega| \leq slope$
    if $\Sigma rowList == row$ **then**
      [t] = perfectSP(t, x_c, τ)
      $start = t$
      set all values in rowList to zero
    else
      $x_c = x_c + \gamma y^*_\omega$
    end if
  end if
end for

Testing top-detection method using simulator

Both these top-detection methods will be tested using the simulator. Two aspects of the top-detection methods, speed and accuracy should be taken into account by the decision which top-detection method is the best. The speed is determined by the amount of oscillation that is needed for the top-detection method to detect a top from the moment that x_c approximates x_{max}. The accuracy is determined in terms of the distance between the expected x_{max} and the real x_{max}. Despite having a clear definition for

both accuracy and speed it is quite difficult to test both these aspects and decide which top-detection method is the best. Something like a weighted average for both the aspects should be used then.

Here is opted for a simplified comparison between the top-detection methods. The cumulative regret at the end of the run will be used here as a quantity of the performance of the top-detection methods. In Equation 1 the formula of regret is given. It is important that the horizon of all the tests is the same. Otherwise, we cannot use the quantity regret to compare them, so the whole run of 10,000 time points will be used during this experiment. If we want to use the whole run we should add a stabilization policy to LiF, which informs LiF if the curve started with drifting. So a *perfect stabilization policy* is used here, the stabilization policy is perfect in the sense that it always knows when the curve is drifting. In Algorithm 8 the pseudo-code of the perfect stabilization policy is given. In Algorithm 7 and Algorithm 5 it is shown how the perfect stabilization policy is called by LiF.

Algorithm 8 Perfect stabilization policy

1: **Funtion** perfectSP(x_c, t, τ)
2: **while** no concept drift at t **and** t $\leq \tau$ **do**
3: $f(x_c) + \epsilon_t$
4: $t = t + 1$
5: **end while**
6: **return** t

So what exactly happens during a run is as follows: LiF starts with searching for x_{max}. If a top-detection method thinks that x_c approximates x_{max}, then the top-detection method detects the top and the perfect stabilization policy will take over the control, until it detects concept drift. The stabilization policy will notify LiF, by giving back the current time point, and LiF will start with oscillation again. before x_c could updated again at least one oscillation should have happened, since the values in y^*_ω are not trustworthy anymore. So if the top-detection methods makes detection during concept drift, this will lead to a higher overall regret.

The advantage of this simplified comparison is that both the performance of the speed and the accuracy are included in the cumulative regret. Namely, a slower top-detection method will result in a higher regret, since more oscillations are needed. Moreover, a top-detection methods with a lower accuracy will lead to higher regret, since the distance between the x_c and x_{max} will be larger when the accuracy is lower. Thus the regret obtained during the period between the top-detection method detects a top and the moment that the stabilization policy notifies LiF that the curve is shifting will be larger. The period is approximate 1000 time points, so you could expect that the accuracy does have a larger influence on the cumulative regret than the speed of the top-detection method, which was requested.

Stabilization policy 1: y-value dependent

The y-value dependent stabilization policy (SP1) uses the observed y-values of the last oscillation to approximate the noise level on the curve and to approximate the y-value at x_c. Given this information SP1 is able to detect concept drift. SP1 only exploits x_c, which was detected as the top by the top-detection method. If the observed y-value significantly differs from the expected y-value given the expected noise, the stabilization policy detects concept drift.

In order to make sure that a single wrong detection would not lead to a notify to LiF, concept drift should be detected as a percentage of the last X steps. The reason why a percentage seems to be better here than just a row, which is used by the top-detection methods, is as follows: let's assume that x_c is close or equal to x_{max}. If concept drift happens the observed y-value at x_c will decrease. Let's assume that the difference between the expected y-value and the observed y-value at x_c is 10. Than the shift is significant, note that noise has standard deviation of 5, and thus the stabilization policy should notify LiF that the curve has shifted. However, in this case only approximately 50% of the time the observations are significantly different from expected, which is the result of the noise on the curve.

The noise on the curve is created with a normal distribution with mean 0 and standard deviation 5. So in approximate 50% of the cases the observed y-value

will be smaller or equal to f(x_c). If this is the case concept drift will still be detected. But in case that the noise is a positive value (not too large), which is also approximate 50% of the time, the observed y-value will not significantly differ from the expected y-value and thus concept drift will not be detected. Using a percentage makes this stabilization policy able to detect concept drift trustworthy even with a small shift. The variable *percentage* was optimized, $percentage \in \{10, 20, 30, 40, 50, 60, 70, 80, 90, 100\}$

To calculate the percentage the last 10 observations are used. Here is decided to use only the last 10 observations since time points far back in the history could give the wrong information. As concept drift could have happened afterwards. The assumption is made here that a length of 10 would be sufficient.

To determine the expected noise and the expected y-value the stabilization policy uses the y-values of the last oscillation. We cannot simply use all the values observed around the expected top during the whole run, since concept drift could have happened. Given the observations of the last oscillation the stabilization-policy tries to fit a 2-degree polynomial through the observed data points. Note that every top of a curve could be approximated by a 2-degree polynomial. So the model of the whole curve should not be known, this is illustrated in Figure 8. The pseudo-code of SP1 is given in Algorithm 10. In Algorithm 9 the stabilization policy is used in LiF in combination with TD2.

An advantage of SP1 is that is does not need oscillation to determine if concept drift happened, therefore the linear regret due to oscillation won't be there.

An disadvantage of SP1 is that it performs less if LiF uses very small amplitude. A large amplitude is needed since the observation of the last oscillation are used to determine the noise level and the expected y-value. If the amplitude of LiF is small, than the last observations were close to each other and it will become hard to fit a trustworthy 2-degree polynomial through the points. Consequently, the noise-level and the expected y-value will be determined badly and the performance of SP1 will decrease.

Another disadvantage of SP1 is that it could not deal with situations where the shape of the curve

Algorithm 9 LiF in combination with TD2 and SP1

Require: x_c, A, T, γ, $\vec{y}_\omega = \{NA_1, ..., NA_T\}$, rowList, row, slope, start = 0, percentage, τ

$\omega = \frac{2\pi}{T}$
for $t = 1, ..., \tau$ **do**
  $x_t = x_c + A\cos\omega t$
  $\vec{x}_t(t \bmod T) = x_t$
  $y_t = f(x_c + A\cos\omega t) + \epsilon_t$
  $\vec{y}_\omega$ = push $(\vec{y}_\omega, y_t \cos\omega t)$
  if $(t > T + start)$ **then**
    $y^*_\omega = \Sigma\vec{y}_\omega / T$
    $rowList(t \bmod row) = |y^*_\omega| \leq slope)$
    if $\Sigma rowList == row$ **then**
      [t] = SP1$(\vec{x}_t, t, x_c, percentage, \tau)$
      $start = t$
      set all values in rowList to zero
    else
      $x_c = x_c + \gamma y^*_\omega$
    end if
  end if
end for

Algorithm 10 SP1

function SP1 $(\vec{x}_t, t, x_c, percentage, \tau)$
$[\beta 1, \beta 2, \beta 3, expectedNoise] \leftarrow$ fit 2-degree polynomial through data samples obtained during last oscillation, so values in $\vec{x}_t$
extremum $\leftarrow$ calculate y-coordinate of extremum of the polynomial with coefficients $\beta 1, \beta 2, \beta 3$
for $t = t, ..., \tau$ **do**
  $y_t = f(x_c) + \epsilon_t$
  $g = \frac{|y_t - extremum|}{expectedNoise}$
  $z(t \bmod 10) = g > 1.96$
  if $\Sigma z \geq percentage$ **then**
    BREAK
  end if
end for
return t

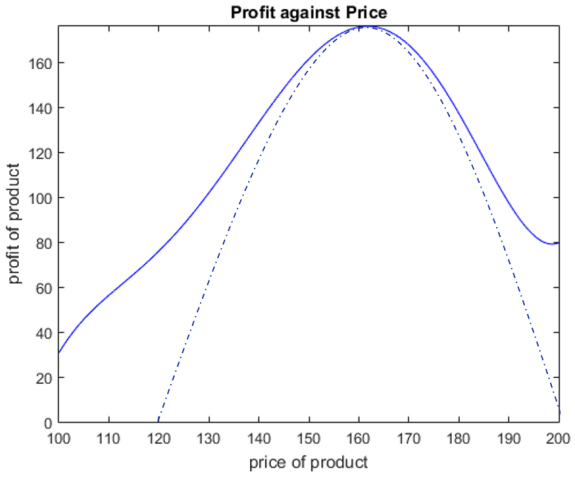

Figure 8: Simplified representation of price-profit curve, where the top approached is by a 2-degree polynomial (dotted-line)

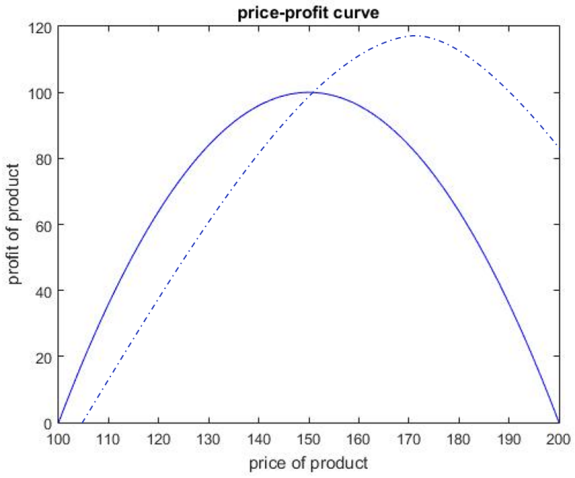

Figure 9: Simplified representation of price-profit curve, where the curve shape changes over time. line = curve at t = 1000, dotted-line = curve at t = 2000

changes over time. An illustration of a situation is given in Figure 9. Before the concept drift x_{max} was 150. After the concept drift and the reshape the x_{max} is close to 170. However, SP1 still observed a profit around 100 at price 150 and thus won't detect concept drift.

Moreover, SP1 works well in situations where the curve is very flat. In this case a small shift won't lead to a significant change in the observed y-value and consequently concept drift will not be detected.

Stabilization policy 2: Slope-dependent

The slope-dependent stabilization policy (SP2) still makes use of oscillation to detect concept drift. However, the oscillation used by SP2 has a smaller amplitude and the amount of oscillation is reduced. A small amplitude is not a problem, since this amplitude is only used to check if the slope of x_c is still flat, thus it is not used to overcome local minimum of flatter parts of the graph.

During this study the amplitude was reduced to $A = 2$ and the time between the oscillations was optimized, $break \in \{50, 100, 150, 200, 250\}$. During this break the policy exploits x_c determined by the top-detection method. After every oscillation the y^*_ω obtained was used to determine if concept drift has happened. If y^*_ω is larger than a predefined slope the policy detects concept drift. This slope was optimized $slope \in \{0.1, 0.3, 0.5, 0.7\}$. It has been decided to use only one oscillation to determine if concept drift did happen, since the noise on the data stream is smoothed quite well by LiF.

The advantage of SP2 compared to SP1 is that this stabilization policy is also able deal with situations where the shape of the curve changes over time. A disadvantage of this method is that it still uses oscillation to detect concept drift.

Additionally, due to the small amplitude and the breaks it could happen that SP2 could not detect concept drift if the slope at the x_c has become almost flat. This could happen if the x_c has set in a local maximum after the drift. In this study this is impossible due to shape of the curve used. The pseudo-code for LiF in combination with TD2 and SP2 is given in Algorithm 11 and the pseudo-code for SP2 is given in Algorithm 12.

Algorithm 11 LiF in combination with TD2 and SP2

Require: x_c, A, T, γ, $\vec{y}_\omega = \{NA_1, ..., NA_T\}$, rowList, row, slope, start = 0, slope, break, τ

1: $\omega = \frac{2\pi}{T}$
2: **for** $t = 1, ..., \tau$ **do**
3: $x_t = x_c + A\cos\omega t$
4: $y_t = f(x_c + A\cos\omega t) + \epsilon_t$
5: $\vec{y}_\omega$ = push $(\vec{y}_\omega, y_t \cos\omega t)$
6: **if** $(t > T + start)$ **then**
7: $y^*_\omega = \Sigma\vec{y}_\omega / T$
8: $rowList(t \bmod row) = |y^*_\omega| \leq slope)$
9: **if** $\Sigma rowList == row$ **then**
10: [t] = SP1$(T, t, x_c, break, slope, \tau, \omega)$
11: $start = t$
12: set all values in rowList to zero
13: **else**
14: $x_c = x_c + \gamma y^*_\omega$
15: **end if**
16: **end if**
17: **end for**

Testing stabilization policies with simulator

To test both stabilization policies, we should make use of a top-detection method. This top-detection method should inform the stabilization policy to start. During this experiment the best top-detection method following the first experiment will be used, where the best can be quantified in terms of regret.

To decide which stabilization policy is the best, we can best quantify in terms of accuracy, where accuracy is defined as:

$$accuracy = \frac{TP + TN}{TN + TP + FP + FN} \tag{3}$$

where TP(TruePositive) stands for the amount of correct detection of concept drift over the whole run. TN(TrueNegative) for the amount of correct detection of no concept drift over the whole run. FP(FalsePositive) for the amount of wrong detection of concept drift over the whole run and FNFalseNegative) for the amount of wrong detection of no concept drift over the whole run. Note

Algorithm 12 SP2

1: **Function** SP2 (T, t, x_c, break, slope, τ, ω)
2: timeLeft = break
3: pauze = **true**
4: **for** $t = t, ..., \tau$ **do**
5: **if** timeLeft ≤ 0 **then**
6: **if** pauze **then**
7: pause = **false**
8: timeLeft = T
9: **else**
10: pause = **true**
11: timeLeft = break
12: **end if**
13: **end if**
14: **if** pause **then**
15: $f(x_c) + \epsilon_t$
16: **else**
17: $x_t = x_c + 2 \times \cos(\omega \times t)$
18: $y_t = f(x_t) + \epsilon_t$
19: $\vec{y}_\omega$ = push $(\vec{y}_\omega, y_t \cos\omega t)$
20: **end if**
21: **if** timeLeft = 1 **and not** pause **then**
22: $y^*_\omega = \Sigma\vec{y}_\omega / T$
23: **if** $|y^*_\omega| \geq slope$ **then**
24: BREAK
25: **end if**
26: **end if**
27: timeLeft = timeLeft - 1
28: **end for**
29: **return** t

that the *perfect stabilization policy* used during the top-detection experiment has a accuracy of 100%.

The amount of possible concept drift detections differs between the stabilization policies. SP2 could only detect concept drift after every break, whereas SP1 could detect concept drift at every time point. Only using the accuracy is not completely fair. Therefore, both the regret and the accuracy will be used here to compare the stabilization policies. The simulator will be used here to keep track of the regret and the accuracy, during the whole run.

Summary of method

In the method the experiments are discussed in detail. In this section the experiments will be summarized in tables to give a clear overview of the whole study. Table 1 gives an overview of the first experiment, where different top-detection methods (TD1 and TD2) in combinations with the perfect stabilization policy are tested.

LiF		Stabilization Policy		
		SP1	**SP2**	**Perfect**
Top-detection	**TD1**	-	-	X
	TD2	-	-	X

Table 1: In this table an overview is given of all possible Top-detection method and Stabilization policy combination. An "X" indicates that the combination is tested during the first experiment, a "-" means that the combination is not tested.

LiF	Stabilization Policy		
	SP1	**SP2**	**perfect**
TD1 or **TD2**	X	X	-

Table 2: In this table an overview is given of all possible Top-detection method and Stabilization policy combination. An "X" indicates that the combination is tested during the second experiment, a "-" means that the combination is not tested. If depends on the results of the first experiment if either TD1 or TD2 is used during the second experiment as the top-detection method

Table 2 gives an overview of the second experiment, where different stabilization policies are tested. The best top-detection method in terms of regret following the first experiment will be used during this experiment. The perfect stabilization policy will not tested here, since this one couldn't be used in practice.

Results

Two different experiments were done in this study, the results of both these experiments will be explained in more detail in the following section.

Top-detection methods

In Figure 10 & 11 the results of the first experiment are shown.

Figure 10 shows the effect of the tuning parameter *row* of TD1. If the instance of *row* is small, smaller than 15, the error-bars are wider. This could be explained by the fact that the instance of the variable *row* is than too small. Consequently, the detections are strongly influenced by the noise and more wrong detections are made, this would make the error-bars wider.

If the instance of *row* is 15 or larger the influence of noise on the detections is reduced and the error-bars become smaller. A possible side effect of increasing the instance of *row* is that it reduces the speed of TD1 and consequently increases the cumulative regret. Because an increasing of the row leads to more oscillation before TD1 could make a detection and these oscillations result in more cumulative regret. This effect could explain the increase of the cumulative regret if the instance of *row* is larger than 20. (see Figure 10)

Given the results, the best instances for *row* seems to be both 15 and 20. To be sure that both instances have the same mean a two-sample t-test is done, as expected the two-sample t-test does not reject the null hypothesis that the two data vectors are from populations with equal means, with $p = 0.9650$. A two-sample t-test makes the assumptions that both

data vectors have the same variance and are normally distributed. A histogram is used to check if the data was normally distributed and Bartlett's test was used to satisfy the assumption that both data vectors did have the same variance. Bartlett's test rejected not the null hypothesis that both settings-combination have the same variance, with ($p = 0.211$).

So there is not just one instance of the variable *row*, which could be seen as the best. Moreover, a larger row does not lead to a lower accuracy, at most for a decreasing of the speed of the TD1. This makes this top-detection method robust, since an erroneously larger selection of the variable *row* does not lead to a much higher overall regret. In real-world situation the perfect instance of the tuning parameter is unknown, so just opt for a large instance of *row* is sufficient to make this top-detection method work.

The results of TD2 are shown in Figure 11. TD2 has two optimized variables, *slope* and *row*. Every bar-group represents a different instance of the variable *slope*. The different colors represent the different instances of the variable *row*, which stands for the amount of detection that should have been happened before the top was definitely detected. On every bar there is a error-bar drawn, which represent the standard deviation over the 50 observations.

Note that there are different settings, where the cumulative regret is equal to 2.3×10^4. This is exactly the cumulative regret of LiF without top-detection method, see Figure 7. So we could conclude here that the TD2 is too strict, it will make no detections at all and the modified LiF just acts as LiF without top-detection method. This does not mean that it is impossible to get a higher cumulative regret than 2.3×10^4 using TD2. Bad initializations of the variables could result in detected tops at positions not close to x_{max}, For example, a *slope* higher than 0.7 in combination with a small instance of row could give this result.

It is remarkable that error-bars dilate if the row-length increases, this is clearly the case when the slope is equal to 0.5 and 0.7. You would expect that an increase of the row-length would lead to less influence of noise and consequently the error-bars would narrow. This effect is already observed in the results of TD1. As apposed to TD1, the error-bars of TD2 increase with an increase of the instance of *row*. An explanation which could explain this phenomena is that TD2 detects relatively fewer tops compared to TD1. So what we are saying is that TD1 is less strict in detecting tops than TD2. So TD1 benefits from a relative large instance of *row*, the effect of the noise is reduced. TD2 on the other hand is very strict in detecting tops and is less influenced by the noise. So TD2 would have less benefit of a relatively large instance of *row*. Making the assumption that x_c is not very stable in the top, this could explain the observed results that the error-bars narrow.

If the slope is smaller this effect is less visible, but it is still there. Only it is weakened due to the fact that the modified LiF becomes more like the normal LiF with a cumulative regret of 2.3×10^4. So the increasing of the cumulative regret, when the row-length is increased could be both explained by a decrease of the speed of the TD2 and the fact that x_c is unstable.

An erroneous selection of both the variables *row* and *slope* could easily lead to a significantly higher regret. In contrast to TD1, TD2 seems to be less robust. However, take in mind that most of these regret could due to the unstable x_c and that a more stable x_c could lead to a more robust TD2. Both the combination *slope* $= 0.5$ and *row* $= 5$ and the combination *slope* $= 0.7$ and *row* $= 0.7$ seems to be the settings with the lowest cumulative regret. Again a two-sample t-test is done to determine if they significantly differ. As expected the two-sample t-test does not reject the null hypothesis that the two data vectors are from populations with equal means, with $p = 0.8114$. A two-sample t-test makes the assumptions that both vector are normally distributed and have the same variance. A histogram is used to check if the data was normally distributed and Bartlett's test was used to satisfy the second assumption that both data vectors did have the same variance. Bartlett's test rejected not the null hypothesis that both settings-combination have the same variance, with ($p = 0.5945$).

Both top-detection methods have several settings that work the best, in terms of regret. So to compare

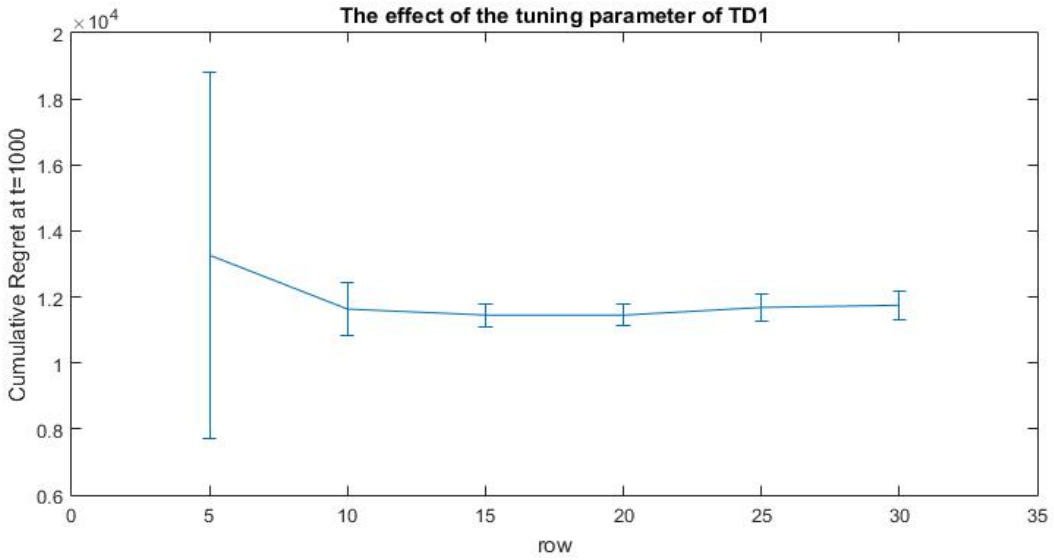

Figure 10: The results obtained during the during the experiment about TD1. The influence of the variable *row* is set against the cumulative regret at $t = 10.000$.

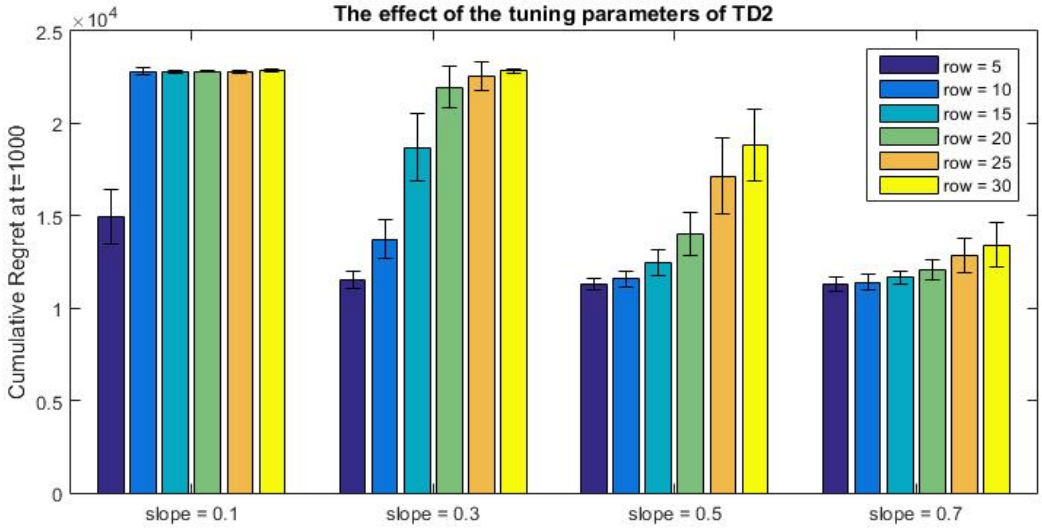

Figure 11: The results obtained during this experiment about TD2. The influence of both the variable *row* and *slope* are set against the cumulative regret at $t = 10,000$. Every bar-group represent a different instance of the variable *slope*. The different colors represent the different instances of the variable *row*

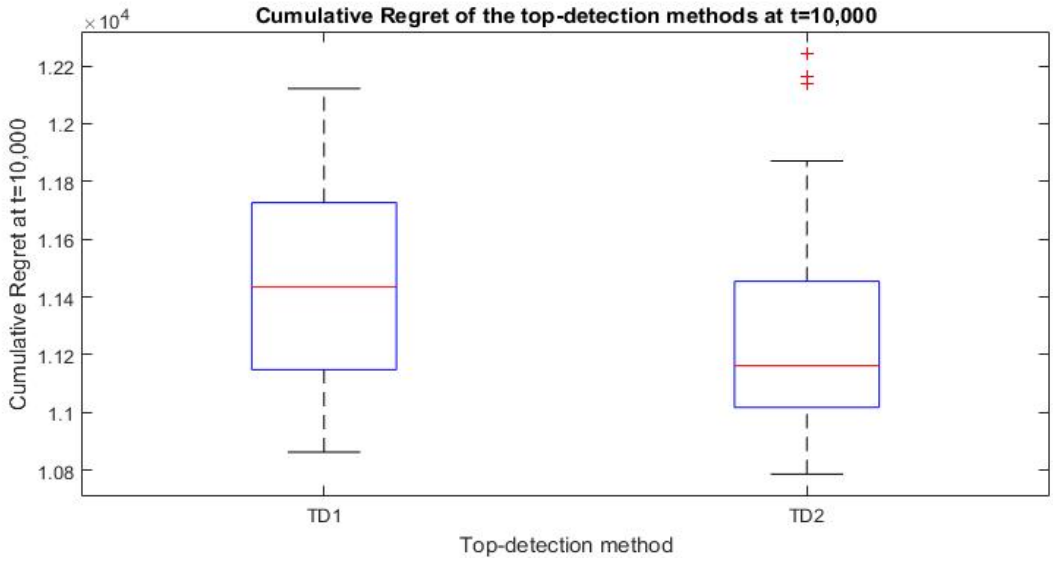

Figure 12: Both the best setting of the top-detection methods compared with each other, where best is quantified in terms of regret. TD2 is better than TD1, with ($p = 0.0200$)

the different top-detection methods, we should choose one of the best settings from both TD1 and TD2. The combination TD1 with $row = 20$ is chosen, since the mean regret was a bit lower, note that this was not significant. The settings for TD2 with the lowest regret was $slope = 0.7$ and $row = 5$. Both the box-plots of the cumulative regret at $t = 10,000$ are shown in Figure 12. TD2 is significantly better, in terms of regret, than TD1, with $p = 0.0200$. This is calculated with the use of a two-sample t-test, the assumption that both vectors have the same variance is satisfied with the Bartlett's test. Bartlett's test rejected not the null hypothesis that both settings-combination have the same variance, with with $p = 0.4333$. The summary of the results is given in Table 3.

During the second experiment TD2 with $row = 5$ and $slope = 0.7$, is used to detect the top. The averaged cumulative regret at t = 10,000 of TD2 in combination with these settings is in combination with the perfect stabilization policy is 1.1289×10^4.

LiF		Stabilization Policy		
		SP1	**SP2**	**perfect**
Top-detection	**TD1**	-	-	1.1452×10^4
	TD2	-	-	1.1289×10^4

Table 3: In this table an overview is given of all possible Top-detection method and Stabilization policy combination. The values represent the observed cumulative regret at $t = 10,000$. A "-" means that the combination hasn't been tested.

Stabilization policies

In Figure 13 and 15 the results for both the stabilization-policies are shown. In these figures only the effect of the the tuning parameters on the cumulative regret is drawn. Besides this also the stabilization policies in terms of accuracy are compared. in the text some of the values will be given, the other accuracies can be found in Appendix A.

Figure 13 shows the results of SP1. This policy only has the variable *percentage* as tuning parameter. On

the x-axis the different instances of the *percentage* are shown. On the y-axis the cumulative regret at t=10,000 is shown. In terms of regret, the best instance of the variable *percentage* = 20%, then the cumulative regret at $t = 10,000$ is $1,1809 \times 10^4$. Note that this cumulative regret is a bit higher than the regret observed if TD2 with the best settings is combined with the perfect stabilization policy (regret is then 1.1289×10^4). It is not surprising that the regret is higher than the regret observed during the first experiment. During that experiment a perfect stabilization policy was used. This stabilization policy had a accuracy of 100%.

The difference between the cumulative regret of TD1 combined with the perfect stabilization policy or TD1 combined with SP1 is quite small. So you would expect that this stabilisation policy is almost perfect. But this is not true, the accuracy of this stabilization policy, with *percentage* = 20%, is 42%. Note that this percentage is very low, since 40% of the run exists of concept drift. There are several reasons possible why this stabilization policy still works fine, in terms or regret, without a high accuracy. To be sure what the exact reason is, we should take a look at the regret over the whole run.

In Figure 14 both the regret of the perfect stabilization policy and SP1 are drawn against the time. Figure 14 makes clear that SP1 is less fast in detecting concept drift, than the perfect stabilization policy. This can be clearly seen between the time points 2000 and 2400. The SP1 starts a bit later with a steeper slope than the perfect stabilization policy, so SP1 detects the concept drift less fast. This effect was expected, since SP1 should first observe abnormal values before it could conclude that the curve has drifted, in contrast to the perfect stabilization policy.

This figure makes also clear that during the drifts SP1 catches up on the perfect stabilization policy. The reason for this should be found in the way LiF is modified. If a stabilization policy detects concept drift, LiF should start with one oscillation before it could update x_c and becomes capable to detect tops. This oscillation is needed to update the values in $\vec{(y)}_\omega$ to make them trustworthy, this extra oscillation will be called *start-oscillation*. This start-oscillation leads to extra regret.

Assume that TD2 does not work that well and detects tops even if the curve is drifting. If TD2 is combined with the perfect stabilization policy, LiF has to make start-oscillation every time TD2 detects a top during a drift. This oscillation starts directly at the time point after that the TD2 has detected the top, since the perfect stabilization policy knows directly that the curve is still drifting. In Figure 14 the regret of TD2 combined with the perfect stabilization policy is quite high during the drifts. This regret could be partly explainable by the fact that the curve is just simply drifting and during drifts it is hard to keep x_c at x_{max}, but a part of this regret could be due to the bad performance of TD2 during the drifts. Beside that it could be that TD2 detects tops even during the drifts, TD2 seems to be quite well in detecting the top if x_c is close to x_{max}. If the curve is not drifting the regret of the TD2 combined with perfect stabilization policy is almost zero.

To go back to the observation that SP1 catches on the perfect stabilization policies during the drift, this could be explained by the assumption made above that TD2 detects tops during the drifts. The perfect stabilization policy would notify LiF directly that the curve is drifting, whereas SP1 would notify LiF with a delay about the fact that the curve is drifting and so fewer start-oscillation are needed during a drift. You could imagine that staying at a certain position will lead to less regret during the drifts than constantly oscillate with an amplitude of 10. This could explain the observed effect that SP1 catches on the perfect stabilization policy during the drifts.

Moreover, Figure 14 makes clear that SP1 loses regret compared to the perfect stabilization policy in case that the curve is not drifting. A reason could be that the accuracy of the stabilization policy is quite low. Even if the curve is not drifting SP1 detects concept drift. A wrong detection of concept drift leads to a start-oscillation, thus every false alarm of the stabilization policy does cost regret.

The best settings for SP1, in terms of the accuracy, are with *percentage* = 10%. The accuracy was somewhere around 42%. The accuracy of SP1 decreases by an increase of the instance of the variable *percentage*. A decrease of the percentage leads to a

decrease of reaction time, because SP1 becomes more strict. Given this observation we could conclude that the accuracy of SP1 is strongly negative influenced by the delay of the detections. The exact obtained data during this experiment are given in appendix A.

In Figure 15 the results of SP2 are shown. In this Figure the bar-groups represent the different instances of the variable *slope*. The different colors represent the different instances of the variable *break*.

This figure makes also clear that an increase of the instance of the variable *break* leads to a higher cumulative regret. This effect is noticeable by most of the slopes, only if the slope is very small an increase of the variable *break* could lead to a decrease of the regret. This effect is understandable if you consider that SP2 with a small slope is stricter than SP2 with a large slope. If SP2 is less strict it is important to check ofter whether concept drift did happen, since the reaction time to concept drifts is slower compared to a stricter variant of SP2. A larger reaction time leads to higher cumulative regret.

Additionally, Figure 15 shows that the standard deviation of the bars increases by the increasing of the instance of the variable *break*. This was as expected. If the break between the concept drift detection possibilities increases, there are less time points to detect concept drift. So these time points become more influential on the cumulative regret. If these time points lay by accident just before the moment that concept drift happens, the overall regret will be larger, compared to the situation where the time points lay exactly after the drift, because the reaction time increases.

The combination $slope = 0.1$ and $break = 150$ and the combination $slope = 0.1$ and break $= 100$ seems to be the best settings for SP2. A two-sample t-test is done to determine if there is a significant difference between the means of these setting-combinations. The two-sample t-test rejects the null hypothesis that both settings-combination have a equal means, with $p = 0.0028$. A special variant of the two-sample t-test is used here, since the variance of both the input vectors are not equal. Since Bartlett's test rejected the null hypothesis that both settings-combination have the same variance, with $p = 0.0026$. So the combination $slope = 0.1$ and $break = 150$ is significantly better than the combination $slope = 0.1$ and $break = 100$ in terms of regret. The cumulative regret at $t = 10,000$ of the best setting-combination is 8.915×10^3.

The cumulative regret at $t = 10,000$ is much lower than the regret obtained during the first experiment where TD2 was combined with the perfect stabilization policy. In Figure 11 the regret over the whole run is drawn. SP2 catches up the perfect stabilization policy during the drifts, just as SP1. The reason for this effect is already explained by results of SP1. Note that the best setting for the slope was, $slope = 0.1$. If we compare this slope with the results obtained during the experiment of the TD2 (see Figure 11), we could conclude that SP2 with a slope of 0.1 would almost always detect concept drift. So given these results you would expect that SP2 is not able to determine in an accurate way whether the curve suffers from concept drift with the given information. So you would not expect that this stabilization policy does have a high accuracy. The results confirm this, SP2 with $slope = 0.1$ and $break = 150$ has a accuracy around 57%. If the slope of SP2 becomes larger the mean accuracy over the different break-lengths decreases.

Comparing both SP1 and SP2 with each other, it is clear that both the best settings of SP1 and SP2 are strict in detecting concept drift. Because of the fact that SP2 makes use of breaks, the amount of possible wrong detections reduces over the whole run. Consequently, fewer start-oscillations are needed and the cumulative regret seems to be significantly lower. A two-sample t-test is done to verify this, the test rejected the null hypothesis that the mean cumulative regret at $t = 10.000$ is equal between the stabilization policies, with $p = 0.00$. A special variant of the two-sample t-test was used here, this variant doesn't make the assumption that the variances of the vectors are equal. Since Bartlett's rejected the null hypothesis that both vectors have the same variance, with $p = 0.0000$. In Figure 17 the box-plots of the stabilization polices with the best settings are drawn. However, SP2 is significantly better in terms of regret than SP1, stating that SP2 has also a higher accuracy is more difficult. The accuracy of SP2 is positively

influenced compared to SP1 by the fact that TD2 detects tops even if the curve is drifting. SP2 checks if concept drift did happen after a break of 150 time points, whereas SP1 does this almost directly. So if the curve is drifting, the effect of the drift during the first possible detection of SP2 is larger than the effect that SP1 deals with. This influenced the accuracy of SP1 in a negative way compared to SP2, so using another top-detection method could give other results. In Table 4 the summary of the results of the stabilization policy are given.

LiF	Stabilization Policy		
	SP1	**SP2**	**perfect**
TD2	$1,1809 \times 10^4$	8.915×10^3	-

Table 4: In this table an overview is given of all possible stabilization policy combinations with TD2. , a "-" means that the combination is not tested. Where the values represent the cumulative regret at $t = 10,000$

Modified LiF compared with the other strategies

So given the results obtained during both the experiments LiF in combination with TD2 and SP2 is the best in terms of regret. So the modified LiF will both make use of TD2 and SP2. This modified LiF will be compared with both Thompson Sampling, ϵ-first and the original LiF to determine how LiF is improved. The 2-degree polynomial used during the experiments, with Equation 2, is used again here to compare the strategies. The only difference is that the curve does not suffer from concept drift during the run. The reason therefore is that both Thompson Sampling and ϵ-first are not able to deal with concept drift.

In Figure 17 the original LiF, the modified LiF, Thompson sampling, and ϵ-first are compared with each other. The cumulative regret of the modified LiF is significantly lower than the regret of the original LiF. However, it suffers still from linear regret. This linear regret is not only due to the small oscillations that SP2 uses, but also due to the fact hat SP2 is actually too strict and detects still concept drift during this run, whereas there is no concept drift at all. During this run only 10% of the detection was correct, which is low but was expected given the results of the second experiment, the settings of SP2 are too strict. These settings works fine in case of a lot of concept drift, but not in case that there is no concept drift at all. Note that TD2 in combination of SP1 would work better here, the accuracy of this combination would be 99.5 %.

Due to this linear regret of SP2 the difference between Thompson Sampling and modified LiF will only increase if the horizon of the simulation is larger. The cumulative regret at $t = 10,000$ of the modified LiF was during this simulation 5.8517×10^3 (mean over 50 runs). Note that the cumulative regret is lower than the regret obtained during the second experiment with the same TD2 and SP2, this reduction is due to the absence of concept drift.

Discussion

The cumulative regret of the modified LiF is significantly lower, compared to the original LiF. The cumulative regret at $t = 10,000$ is reduced from 2.3×10^4 to 5.8517×10^3 (situation without concept drift), which is a large reduction. But note that a very large amplitude ($A = 10$) was used during this study, whereas a smaller amplitude was also sufficient for the model used (Equation 2), because no local maximum or flatter parts should be overcome during the search to x_{max}. The use of a smaller amplitude would also reduce the cumulative regret of the original LiF. So if the best settings for the original LiF where compared with the modified LiF the reduction of the regret would probably be smaller.

During the first experiment a perfect stabilization policy was used to compare both the top-detection methods with different settings with each other. In contrast to the second experiment, where TD2 was used, which is not a perfect top-detection method. A perfect top-detection method would only detect tops if x_c is very close to x_{max} (difference smaller than 0.5) and if the curve is not drifting, TD2 does not satisfy this assumption. Especially the fact that TD2

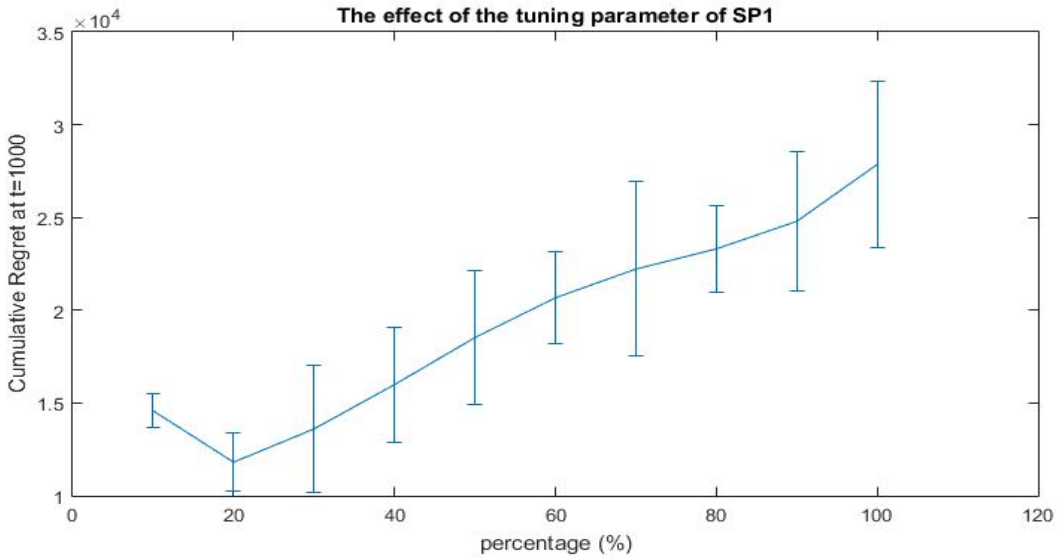

Figure 13: The influence of the variable *percentage* is drawn against the cumulative regret at $t = 10,000$ of SP1

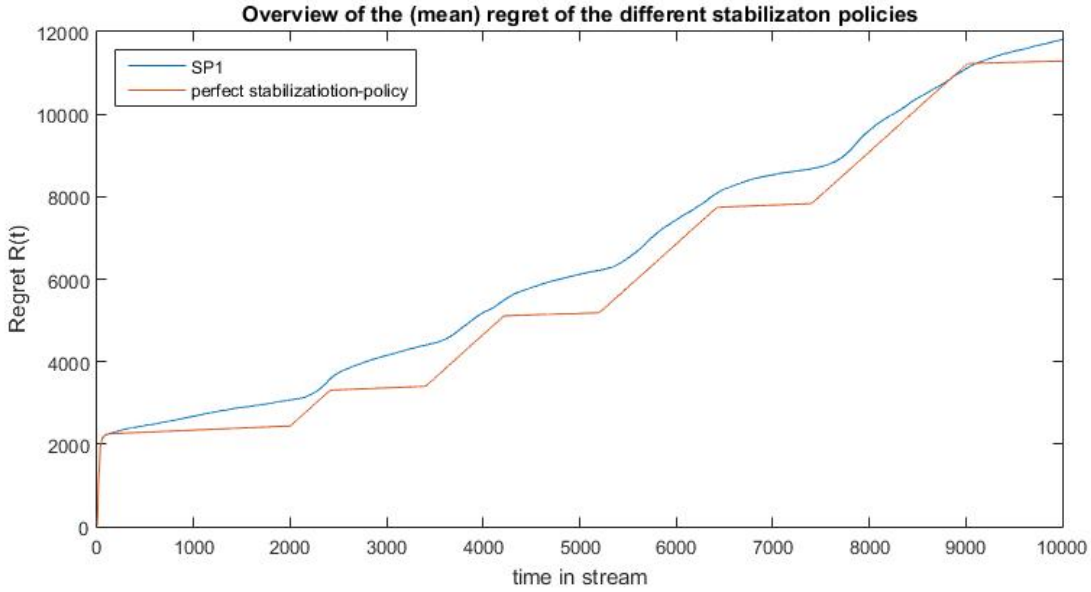

Figure 14: An overview of the cumulative regret over the whole run. Both for combinations TD2 + SP1 and TD2 + perfect stabilization policy

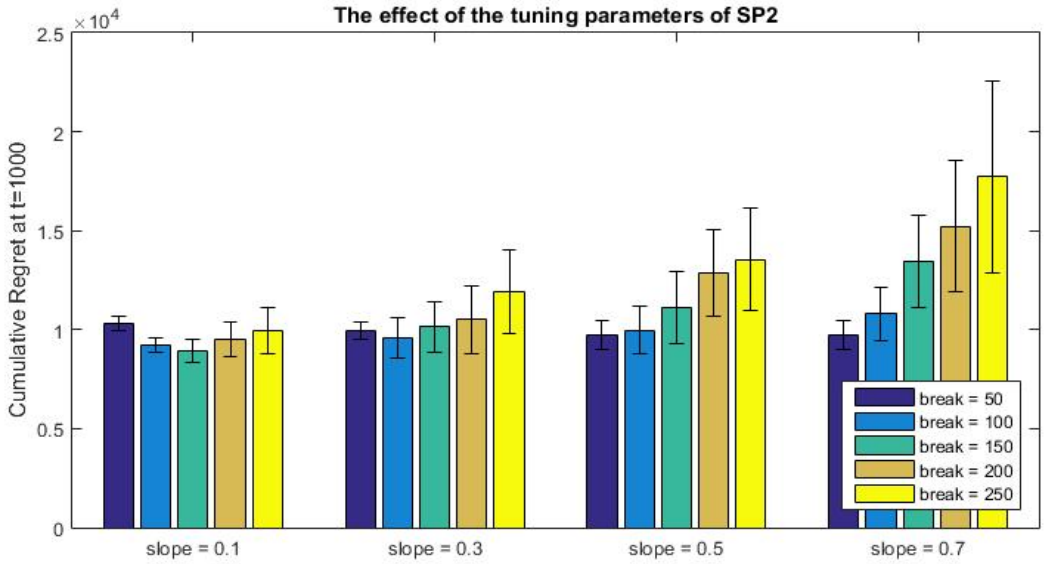

Figure 15: The influence of the variable *slope* and *break* on the cumulative regret at $t = 10,000$ of SP2

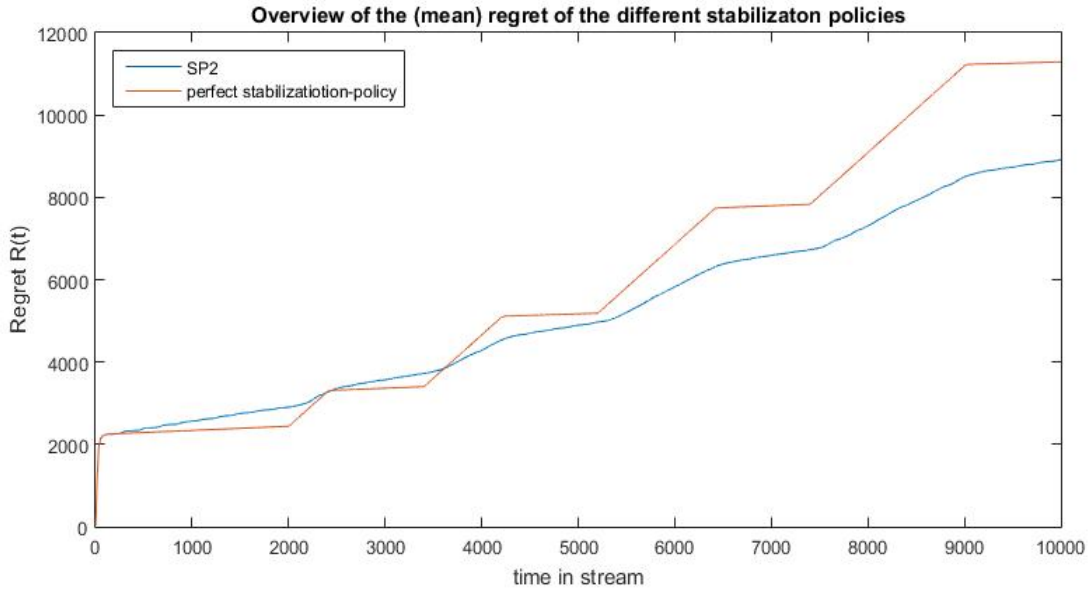

Figure 16: An overview of the cumulative regret over the whole run. Both for combinations TD2 + SP2 and TD2 + perfect stabilization policy

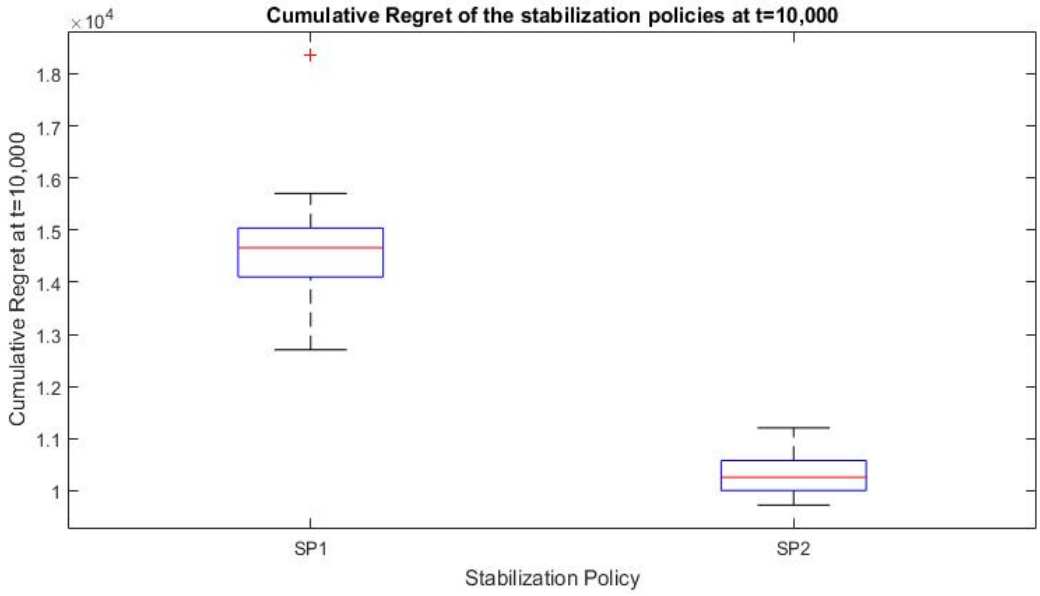

Figure 17: Both the best setting of the stabilization compared with each other, where best is quantified in terms of regret. SP2 is significantly better than SP1. ($p = 0.00$)

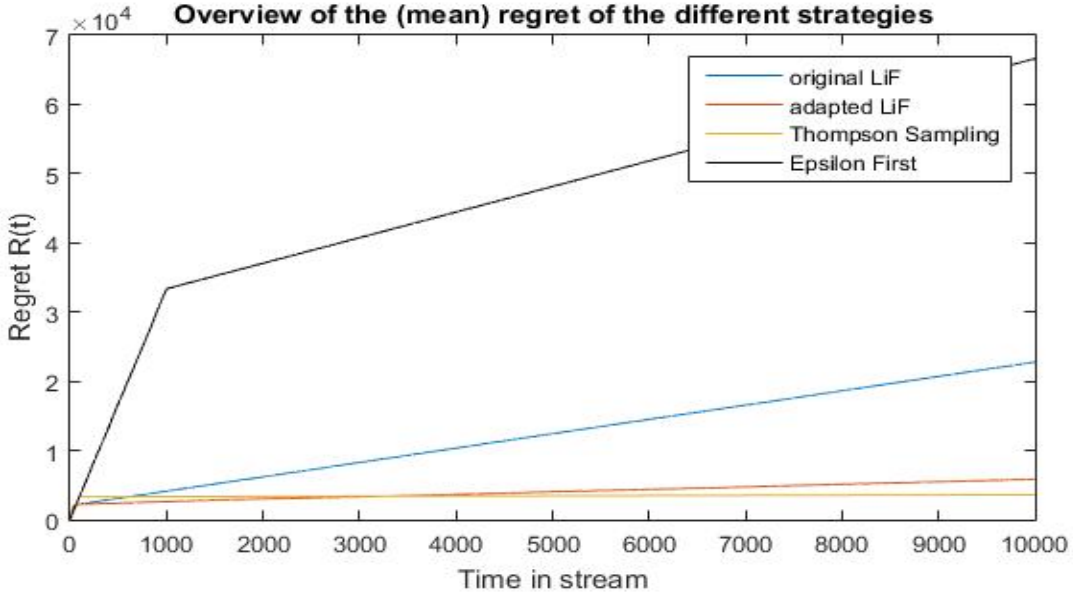

Figure 18: The four different strategies: Thompson Sampling, ϵ-first, original LiF, and modified LiF are compared with each other over the whole run in terms of the cumulative regret

detects tops even if the curve is drifting could have influenced the accuracy of strict stabilization policies in a positive way. To improve this study a perfect top-detection method should be used during the second experiment.

Finally, only a simplified model (Equation 2) was used during this study. No further research was done to real-world price-profit curves and the amount of concept drift on these curves. So the obtained best instances of the examined variables are not guaranteed to work in real-world problems. More research should be done to real-world shapes of price-profit curves and the amount of concept drift these curves suffer from. Such that a simulator could be made, which generates real-world price-profit curves, to determine which settings would work for these problems.

Conclusion

The aim of this study was to answer the question: Could we adapt the policy of LiF such that it needs fewer oscillations to detect concept drift in order to reduce its linear regret?

To answer this question two different simulation studies have been done. During the first experiment two different top-detection methods were compared with each other, TD1 and TD2. The aim of the top-detection methods is to detect if x_c is close to x_{max}. If x_c is close the x_{max} LiF only uses the oscillations only to detect concept drift and not for the search anymore. The assumption is made that the oscillations are very important during the search to find x_{max}, but less important if they are only used to detect concept drift. So another policy (stabilization policy) could do this work, with the use of no or fewer oscillations than LiF would use in order to reduce LiF linear regret. To be able to know when the stabilization policy should start the top-detection method should detect this situation. During the second experiment two stabilization policies are compared with each other SP1 and SP2. Both the stabilization polices and the top-detection methods are compared with each other in terms of the cumulative regret.

During the first simulation study the top-detection methods where compared with each other. To be able to do a simulation study TD2 and TD1 ware combined with a perfect stabilization policy, perfect in the sense that is knows exactly when the curve was drifting or not. Both top-detection methods have several optimizable variables, firstly the variables where optimized with the use of the simulator and subsequently the best setting for the top-detection methods were compared with each other. In Table 5 the results of both the best setting of the top-detection methods are given in column with the header "perfect". Although TD2 was significantly better than TD1 in terms of regret, still TD2 does have some characteristics which were not desired. TD2 makes detections even if the curve was drifting, this behaviour leads to a higher cumulative regret. Despite this, if the curve was not drifting the accuracy of the detection was high, in the sense that x_c was close to x_{max}.

During the second simulation study both the stabilization policies were compared with each other. To be able to do a simulations study SP1 and SP2 where combined with TD2, the best top-detection method following the first experiment in terms of regret. Both stabilization polices have several optimizable variables. Firstly, the variables were optimized for both these stabilization policies in terms of regret, subsequently both the stabilization policies with the best settings where compared with each other. So the same procedure as during the first experiment. The results of the simulation study show that SP2 was better in terms of regret than SP1. However, the accuracy of both the stabilization policies was very low. Despite this, the cumulative regret of SP2 was still lower than the regret observed if TD2 was combined with the perfect stabilization policy, this could be explained by the fact that TD2 makes detections even if the curve was drifting. Both TD2 and SP2 uses the approximate slope by LiF to make a detection, it seems that this approximate slope by LiF is a useful information that could be used to improve LiF.

So to answer the research question, it is possible, with the examined stabilization policies to adapt the pol-

LiF		Stabilization Policy		
		SP1	**SP2**	**perfect**
Top-detection	**TD1**	-	-	1.1452
	TD2	1,1809	0.8915	1.1289

Table 5: In this table an overview is given of all possible top-detection method and stabilization policy combinations. An "-" means that the combination is not tested. The values in the table represent the cumulative regret at $t = 10,000$. All the values in the table should be multiplied with 10^4

icy of LiF in such a way that it needs less oscillation to detect concept drift, both stabilization policies were able to detect concept drift. Unfortunately, the accuracy of both SP1 and SP2 was low. Despite this, LiF combining with TD2 and SP2 makes LiF more robust. During the search a very large amplitude can be used, the disadvantage of a large amplitude during the exploit phase is reduced by TD2 and SP2. In simple models without local maxima and flatter parts if is probably better to use LiF with a small amplitude in order to get the lowest regret. Note that it is hard to determine if the model is simple enough to use either LiF or the modified LiF in practice.

Given all the results I would suggest the following adaptations for LiF as long as there is no stabilization policy with a higher accuracy. Combine LiF only with a top-detection method, for example TD2, to determine if x_c is close to x_{max}. During the search LiF could make use of a large amplitude to overcome local maxima and flatter parts and after the top has been detected the amplitude of LiF could be reduced to a smaller amplitude. Such that LiF benefits from the fact that is uses a large amplitude during the search and it has less disadvantage of the regret produced by oscillation during the exploitation phase, compared to the situation where the amplitude is still large.

Further Work

Firstly, more research should be done to other stabilization policies or to improve the stabilization policies introduced in this study. The accuracy of SP2 could for example be improved by using a higher integration time. Using a higher integration time would give more observations per oscillation and consequently the influence of the noise is more reduced, this could give SP2 a more trustworthy observation of the slope of x_c.

If a higher integration time is key to improve SP2, you could even think about more improvements. The breaks of SP2 are now optimized for the whole run and are not dynamic. In real-world situation the optimal size of the breaks is unknown and to make LiF more robust you should opt for a small break, consequently SP2 starts to behave as LiF without stabilization policy. To make SP2 more robust a dynamic break could be used. This dynamic break could be increased if concept drift isn't detected several times in a row. Only a maximum value for the break should be internalized then, to prevent that the break becomes too large and stabilization policy is not able to detect the drift anymore. Another option could be to combine SP1 and SP2, if the breaks are too large SP1 could be used to detect concept drift.

Secondly the top detection methods could be optimized in terms of accuracy. During this research the top-detection methods did not make use of any further information than the information obtained during the oscillation of LiF. This was done to keep the cumulative regret as low as possible. But it seems that the top-detection methods did not work very well and did also detect tops if the curve was shifting. So maybe it is better to invest more, for example by observing more x-values, such that the top-detection method will become more accurate. Especially in curves with less concept drift, a more precise top-detection method which uses more observations to detect the top would give at the end a lower cumulative regret. You could for example think about combining two top-detection methods, the first top detection is a rough detection of the top and the second top detection is more precise, by using for example a higher integration time.

Finally, the results of SP1 showed that, you can fairly easy approximate the noise on the curve. The averaged noise on the curve approximated by SP1 was expected to be 5.2, whereas the real noise on the curve had standard deviation of 5. This approxima-

tion of the noise is close enough to use it for improving LiF. Namely, if the noise level on the curve is large, a larger integration time is needed. Whereas a small integration time is already enough if the noise level is low. So if the noise is approximated at the start of the simulation the integration time could be optimized for that noise level. In case LiF with the batch approach is used, the integration time could even be dynamically updated during the run.

References

[1] Rajeev Agrawal. The continuum-armed bandit problem. *SIAM Journal on Control and Optimization*, 33(6):1926–1951, July 1995.

[2] Paul Benjamin. Human-inspired algorithms for search: A framework for human-machine multi-armed bandit problems. *Mechanical and Aerospace Engineering*, September 2014.

[3] Niall M. Adams Nicos G. Pavlidis & David J. Hand Christoforos Anagnostopoulos, Dimitrsi K. Tasaulis. Online linear and quadratic discriminant analysis with adaptive forgetting for streaming classification. *Statistical Analyses and Data Mining*, 5(2):139–166, April 2012.

[4] J.C Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society*, 41(2):148–177, 1979.

[5] Shipra Agrawal & Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. *Proceeding of the 30th International Conference on Machine Learning*, 28(3):127–135, June 2013.

[6] Maurtis Kaptein & Davide Iannuzzi. Lock in feedback in sequential experiment. Februari 2015.

[7] Dean Eckles & Maurits Kaptein. Thompson sampling with the online bootstrap. October 2014.

[8] Robert Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. *Advances in Neural Information Processing systems 17*, pages 697–704, 2004.

[9] Olivier Chappelle & Lihong Li. An empirical evaluation of thompson sampling. *JMLR:Workshop and Conference Proceedings*, 23(39):1–26, 2012.

[10] R.E. McInerney. Multi-armed bandit bayesian decision making. 2010. PhD Thesis.

[11] Mike Meade. Lock-in amplifiers: principels and applications. 1, 1983.

[12] Joannès Vermorel & Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. *16th European Conference on Machine Learning*, pages 437–448, October 2005.

[13] Ronald Ortner & Csaba Szepesvári Peter Auer. Improved rates for the stochastic continuum-armed bandit problem. *Lecture Notes in Computer Science*, 4539:454–468, June 2007.

[14] Volodymyr Kuleshov & Doina Precup. Algorithms for multi-armed bandit problem. *Journal of Machine learning Research 1*, pages 1–48, October 2000.

[15] Joseph Mellor & Jonathan Shaprio. Thompson sampling in a switching environments with bayesian online change point detection. *AIStats 2013*, Februari 2013.

[16] Navin Goyal Shipra Agrawal. Analysis of thompson sampling for the multi-armed bandit problem. April 2012.

[17] Andreas Krauze & Joel Burdick Thomas Desautels. Parallizing exploration-exploitation tradeoffs with guassian process bandit optimization. *The Journal of Machine Learning Research*, 15(1):3873–3923, Januari 2014.

[18] Eric W.Cope. Regret and convergence bounds for a class of continuum-armed bandit problems. *IEEE Transaction on Automatic Control*, 54(6):1234–1253, June 2009.

[19] Willem G. Macreday & David H. Wolpert. Bandit problems and the exploitation/exploitation tradeoff. *IEEE Transactions on Evolutionary Computation*, 2(1):2–22, April 1998.

A Data

In this appendix the most important data obtained during this research are given. The first two tables, Table A1 and Table A2, show the results of both the top-detection methods TD1 and TD2. The latter two tables, Table A3 and Table A4, show the results of both the stabilization policies SP1 and SP2. The following abbreviations are used in the tables:

- CR = cumulative regreat at $t = 10,000$
- LCB = 95% Lower confindence bound
- UCB = 95% Upper confidence bound

Row	**CR**	**LCB**	**UCB**
5	1.3265×10^4	1.1685×10^4	1.4845×10^4
10	1.1635×10^4	1.1405×10^4	1.1864×10^4
15	1.1455×10^4	1.1357×10^4	1.1552×10^4
20	1.1452×10^4	1.1357×10^4	1.1547×10^4
25	1.1684×10^4	1.1570×10^4	1.1789×10^4
30	1.1752×10^4	1.1627×10^4	1.1877×10^4

Table A1: Table with data of the experiment 1; TD1

Row	slope	CR	LCB	UCB
5	0.1	1.4935×10^4	1.4532×10^4	1.5373×10^4
5	0.3	1.1509×10^4	1.1374×10^4	1.1645×10^4
5	0.5	1.1306×10^4	1.1212×10^4	1.1400×10^4
5	0.7	1.1289×10^4	1.1188×10^4	1.1391×10^4
10	0.1	2.2813×10^4	2.2765×10^4	2.2870×10^4
10	0.3	1.3741×10^4	1.3449×10^4	1.4034×10^4
10	0.5	1.1591×10^4	1.1472×10^4	1.1711×10^4
10	0.7	1.1395×10^4	1.1279×10^4	1.1512×10^4
15	0.1	2.2815×10^4	2.2793×10^4	2.2838×10^4
15	0.3	1.8707×10^4	1.8195×10^4	1.9219×10^4
15	0.5	1.2451×10^4	1.2259×10^4	1.2644×10^4
15	0.7	1.1661×10^4	1.1561×10^4	1.1761×10^4
20	0.1	2.2822×10^4	2.2804×10^4	2.2839×10^4
20	0.3	2.1965×10^4	2.1642×10^4	2.2287×10^4
20	0.5	1.3989×10^4	1.3661×10^4	1.4318×10^4
20	0.7	1.2087×10^4	1.1931×10^4	1.2243×10^4
25	0.1	2.2816×10^4	2.2796×10^4	2.2836×10^4
25	0.3	2.2560×10^4	2.2343×10^4	2.2777×10^4
25	0.5	1.7137×10^4	1.6549×10^4	1.7724×10^4
25	0.7	1.2825×10^4	1.2558×10^4	1.3092×10^4
30	0.1	2.2847×10^4	2.2826×10^4	2.2867×10^4
30	0.3	2.2834×10^4	2.2809×10^4	2.2858×10^4
30	0.5	1.8853×10^4	1.8302×10^4	1.9404×10^4
30	0.7	1.3400×10^4	1.3059×10^4	1.3742×10^4

Table A2: Table with data of experiment 1; TD2

Percentage (%)	CR	LCB	UCB	Expected Noise	accuracy
10	1.4578×10^4	1.4320×10^4	1.4837×10^4	5.26	0.4243
20	1.1809×10^4	1.1364×10^4	1.2254×10^4	5.25	0.4168
30	1.3602×10^4	1.2633×10^4	1.4571×10^4	5.29	0.4163
40	1.5990×10^4	1.5114×10^4	1.6867×10^4	5.21	0.4126
50	2.8526×10^4	1.7492×10^4	1.9559×10^4	5.21	0.4121
60	2.0670×10^4	1.9963×10^4	2.1377×10^4	5.27	0.4098
70	2.2217×10^4	2.0881×10^4	2.3553×10^4	5.16	0.4106
80	2.3306×10^4	2.2635×10^4	2.3977×10^4	5.14	0.4096
90	2.4792×10^4	2.2371×10^4	2.5868×10^4	5.17	0.4090
100	2.7856×10^4	2.6591×10^4	2.9121×10^4	5.28	0.4070

Table A3: Table with data of experiment 2; SP1

break	slope	CR	LCB	UCB	accuracy
50	0.1	1.0297×10^4	1.0195×10^4	1.0400×10^4	0.5658
50	0.3	9.9615×10^4	9.8349×10^3	1.0088×10^4	0.5396
50	0.5	9.7378×10^4	9.5342×10^3	9.9413×10^3	0.5208
50	0.7	9.7262×10^4	9.5250×10^3	9.9273×10^3	0.4948
100	0.1	9.2195×10^4	9.1122×10^3	9.3268×10^3	0.5715
100	0.3	9.6020×10^4	9.3164×10^3	9.8897×10^3	0.5348
100	0.5	9.9572×10^4	9.6172×10^3	1.0297×10^4	0.5283
100	0.7	1.0806×10^4	1.0420×10^4	1.1192×10^4	0.5161
150	0.1	8.9158×10^4	8.7493×10^3	9.0824×10^3	0.5720
150	0.3	1.0145×10^4	9.7770×10^3	1.0512×10^4	0.5600
150	0.5	1.1116×10^4	1.0590×10^4	1.1642×10^4	0.5235
150	0.7	1.3437×10^4	1.2768×10^4	1.4105×10^4	0.5188
200	0.1	9.5217×10^4	9.2730×10^3	9.7705×10^3	0.5872
200	0.3	1.0507×10^4	1.0024×10^4	1.0990×10^4	0.5664
200	0.5	1.2855×10^4	1.2228×10^4	1.3481×10^4	0.5353
200	0.7	1.5228×10^4	1.4284×10^4	1.6173×10^4	0.5133
250	0.1	9.9483×10^4	9.6251×10^3	1.0272×10^4	0.5744
250	0.3	1.1922×10^4	1.1325×10^4	1.2519×10^4	0.5441
250	0.5	1.3550×10^4	1.2810×10^4	1.4290×10^4	0.5701
250	0.7	1.7736×10^4	1.6360×10^4	1.9111×10^4	0.5079

Table A4: Table with data of experiment 1; SP2