Online One-Way Trading with Machine Learned Advice.
MSc Thesis

Mark Heinsbroek
m.g.q.heinsbroek@students.uu.nl
Supervisor: Alison Liu & Jonathan Toole-Charignon
Algorithms and Complexity Group
University of Utrecht

August 18, 2022

Abstract

In the one-way trading problem, a player is presented with a number of rates. For each rate the player needs to decide if they want to trade, and how much of their budget they want to trade for a profit. This trade will be done according to the rate currently available to the player. The goal of the player is to maximise the profit they make on the sum of all trades. The discussed algorithm for one-way trading uses a semi-online environment. More specifically, the number of rates that will be presented is known and the bounds on the minimum and the maximum exchange rates are also known.

In this work, the online one-way trading algorithm is extended so that it can use a piece of advice that is given before the algorithm starts. This advice will influence the performance of the algorithm. More precisely, the effect on the performance will depend on the error of the advice and on the trust that is put in the advice. With advice that has no error, the algorithm can achieve the optimal amount of profit. Depending on how big the error is, the algorithm that uses advice can perform worse than the original online algorithm that doesn't use advice. In this work, two different kinds of advice are tested. The best day to trade on and the highest rate that will be presented.

With the advice tested in this work, one-way trading with perfect advice will always be as good or better than one-way trading without advice. When the presented number of rates is large the online algorithm with advice can outperform the standard online algorithm, even with a significant error in the advice. This is especially true for the advice that predicts the best day to trade on.

1 Introduction

Algorithms are commonly used to make decisions based on the available information. However, in many cases this information might not be complete, especially when making decisions about the future. Algorithms that have to make decisions based on incomplete information are called **online algorithms**. Information is given to the online algorithm piece by piece in a serial fashion. After new information is revealed the online algorithm has to make a decision, this decision is irrevocable.

To compare different online algorithms for the same problem the **competitive ratio** is used. The competitive ratio measures how well an online algorithm performs compared to the **offline optimal solution**. The optimal solution has access to all of the information, including future information that is not available to the online algorithm. Online algorithms can never be better than the optimal solution, otherwise the optimal solution would not be optimal. The competitive ratio is an upper bound on the ratio between the optimal solution and the solution of the online algorithm on all possible input instances I.

$$\frac{\mathrm{ALG}(I)}{\mathrm{OPT}(I)} \leq c$$

This competitive ratio is for minimisation problems. A competitive ratio of two means that no matter what the input instance is the online algorithm will never be more than two times worse than the optimal solution. The online algorithm can be less than two times worse for some input instances, but will never be more than two times worse. The competitive ratio will never be smaller than one, since this would mean that the online algorithm is better than the optimal solution. The competitive ratio is calculated differently for maximisation problems.

$$\frac{\mathrm{OPT}(I)}{\mathrm{ALG}(I)} \leq c$$

With a competitive ratio of two the online algorithm will still never be more than two times worse than the optimal solution. But now the online algorithm will have a lower result than the optimal. When using the competitive ratio for comparing online algorithms the goal is to make the competitive ratio as close to one as possible.

Financial analysis and online algorithms go well together since these problems are often online in nature. In the buy or rent problem, a player is going to use something for an unknown number of days. Every day the player is presented with a choice. The player either buys for a fixed cost B, or keep renting for cost R per day. The player can rent for as long as they want and buy on any day. Depending on how long the player is going to rent it might be cheaper to buy instead. The optimal solution is to check if R times the number of days is more than B. If buying will cost less then the optimal solution buys on day one for cost B, otherwise the optimal solution rents. The online algorithm does not have this luxury and will most likely pay more than the optimal.

2 Online Trading Algorithms

The online one-way trading problem is defined as follows. A player starts with a **budget**, which the player wants to trade into a **profit**. At each time step an **exchange rate** is provided at which the player can trade any fraction of their budget for profit at the given exchange rate. Since this is an online problem the player only knows the revealed exchange rates, and any decision made can't be reversed. The goal of the player is to trade their starting budget for the maximum amount of profit. In this work only the profit will be counted, any remaining budget is irrelevant. If there is any budget left on the last exchange opportunity this will all be spent using the last rate.

2.1 Semi-Online Environment

To measure the performance of an online algorithm it is compared to the optimal solution. This can lead to some issues when using a purely online environment, in which nothing is known by the online algorithm. If the number of exchange opportunities are unknown, then the online algorithm would have a hard time choosing when to spend the budget. There might be only one trading opportunity, so if the algorithm doesn't trade anything on the first day then the resulting profit would be 0. Since the optimal solution will trade their budget the resulting competitive ratio is $\frac{\text{OPT}}{0}$. A solution that isn't very useful.

If there are many trading days, there is still an issue since the online algorithm is pressured to trade early to prevent any budget remaining after the last trading day. Any of the first trading days can be a fraction of the rate that the optimal solution will take. There can be an infinite amount of trading days, at some point the algorithm is bound to spend all of it's budget. Eventually some arbitrarily high exchange rate can be presented. The optimal solution will take this rate, and the competitive ratio will be $\frac{\infty}{\text{ALG}}$.

These examples show that to talk about solution to the online one-way trading problem certain limitations have to be put in place. The second example shows a second limitation of a purely online environment. If the online algorithm spends most of its budget before the last trading day, the last day can still have an arbitrarily large exchange rate. Again, the competitive ratio is $\frac{\infty}{\text{ALG}}$. This means an upper limit is needed on the possible exchange rates. The same is true with exchange rates that get infinitely close to 0, so a minimum exchange rate also needs to be determined.

In this paper, an algorithm proposed by El-Yaniv et al. [4] will be used. This algorithm uses a known amount of trading days n and has a minimum and maximum bound on the possible exchange rates, m and M, respectively. The minimum and maximum exchange rates may not be tight bounds. This means that the presented exchange rates have to be between m and M but don't have to include these bounds. This means that the algorithm is not a pure online algorithm, but a **semi-online algorithm**.

3 Threat Based Approach

The algorithm that El-Yaniv et al. [4] proposed has a variable competitive ratio c. This algorithm relies on c being an achievable competitive ratio, but not all c are achievable. If c is not achievable the algorithm will spend parts of the budget on rates that are too low. The resulting competitive ratio will be higher than c. To make calculating an achievable c possible, the algorithm follows two rules.

1. Only trade on days when the current rate is a new maximum.
2. Only convert the minimum amount to maintain a competitive ratio of c.

These two rules apply to all but the last trading day; on the last trading day all of the remaining budget is traded since any remaining budget could have been turned into more profit. Section 3.2 shows how to get an achievable c. For now it is simply assumed that c is an achievable competitive ratio.

3.1 Optimal "Threat Based" Policy for One-Way Trading

The threat based approach works by making the online algorithm operate under a realistic assumption about future exchange rates. In the semi-online model, the exchange rates are still unknown, but not completely unknown since exchange rates have an upper and lower bound.

To achieve a competitive ratio of c, the algorithm is threatened by the scenario where exchange rate dropping to the minimum rate for all future rates. If you have a series of n days on which you can exchange once per day, these n days can be divided into two parts. A series of k consecutive days where the exchange rate goes up every day, meaning that every day is a new maximum. Then after day k, the rest of the rates will all be the minimum m until the last day n with $k \leq n$.

Since the first day is always a new maximum, according to rule 1 the algorithm will want trade on this day. Since it knows the minimum possible rate, the algorithm can calculate what the resulting competitive ratio is if trades some it the budget. The profit will always be at least the current profit plus all of the remaining budget traded at the minimum rate. If the exchange rate on the first day is higher than the minimum, then the achieved competitive ratio will be lowered by trading right now. When using this strategy, a maximum amount of budget can be saved to spend on higher maximal values that might be presented in the future. Only if the presented rate is the highest possible rate should the online algorithm spend all of its budget.

The optimal solution gets the largest profit by trading on the day with the highest exchange rates. The online algorithm can also trade at this exchange rate though. The worst case scenario is when the algorithm has nothing left to trade when this maximum is reached. To reduce the budget as much as possible only new maximal rates should be presented. These rates should be smaller than the rate the optimal solution will trade at when possible. If a new

maximal exchange rate is presented, this will always increase the profit that the optimal solution will make. The player can only increase their profit if they save more of their budget for this rate.

Let r be a competitive ratio that the online algorithm can achieve. That is, $\mathrm{OPT}(I)/\mathrm{ALG}(I) \leq r$ for all possible input instances I. When a viable r is known the algorithm is clear, simply trade just enough to maintain r if all future exchange rates are equal to m. The amount that needs to be traded s_i is dependant on the budget before making the trade X_{i-1}, the profit before making the trade Y_{i-1}, the current given exchange rate p_i and the threat m. The amount can be calculated according to (1)[4].

$$s_i = \frac{p_i - r * (\mathrm{Y}_{i-1} + \mathrm{X}_{i-1} * m)}{r * (p_i - m)} \tag{1}$$

To find r we look at the worst sequence of exchange rates. The sequence where the ratio between OPT and ALG is maximised. The sequence of rates needs the length of the series of consecutive maximal values k, the smallest presented rate and the biggest presented rate. We need to determine the worst values these variables can take. The smallest rate is at least m and the biggest rate is at most M.

The player would normally only know the competitive ratio after the algorithm is done. Since the competitive ratio needs to be known beforehand it has to be calculated. El-Yaniv et al. [4] provide a way to calculate this r in such a way that r is minimal but will still always be a viable competitive ratio (2).

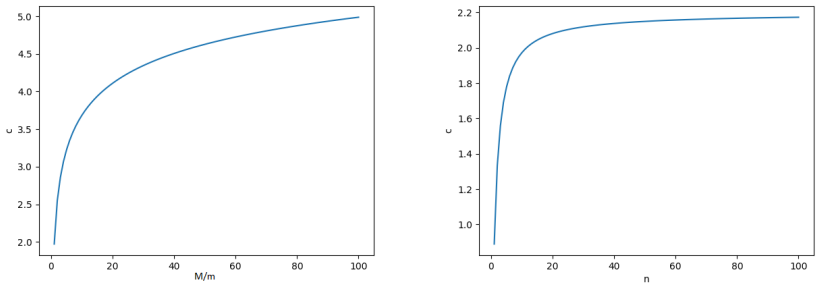
$$r = n * \left(1 - \frac{m(r-1)}{M-m}\right)^{(1/n)} \tag{2}$$

3.2 Calculating an Achievable Competitive Ratio c

The variables m, M and n are known, where $m < M$ and k is the number of days that have consecutive maximal values with $k \leq n$ and k is unknown to the player. The proof relies on the worst values the smallest presented exchange rate, the biggest presented exchange rate and k can take. If all of these worst values are known and we can calculate the competitive ratio possible on these values, we will have an achievable competitive ratio. After minimising this ratio the best possible, achievable, ratio c has been found.

We know how much the player will spend each day, though this depends on r. Since the entire budget will be spend on day n the total spending is also known. The remaining variables are biggest exchange rate, the smallest exchange rate and k. Figure 1a shows that as the difference between m and M goes up, so does the competitive ratio, the same is true for n (Figure 1b). The full proof can be seen in [4].

The competitive ratio will be the worst when the difference between the smallest and largest values is the biggest, and when there are as many consecutive maximal values as possible. This gives us a smallest exchange rate of m,



(a) c goes up as M/m goes up with a fixed n

(b) c goes up as n goes up with a fixed M/m

Figure 1: Limits of n and M/m.

biggest exchange rate of M and $k = n$ as the worst input instance of consecutive maximum values.

4 Advice for online algorithms

Advice given to the online algorithm is information about the future. The goal of using advice is to improve the competitive ratio [2]. How the advice impacts the competitive ratio is dependant on the quality of the advice. If the advice is perfect an online algorithm can perform as well as the optimal solution, depending on what the advice is. However, assuming that the advice is perfect is not realistic. Even if the advice is assumed to be trustworthy, it could still give wrong advice sometimes. Especially when the advice comes from machine learning[3].

Depending on how reliable the given advice is, the algorithm might be better off without advice. Following bad advice can result in a much higher competitive ratio. It is impossible for the online algorithm to know if the advice is correct or not until the prediction either comes true or it doesn't. When designing the algorithm, the possibility of bad advice should be taken into account. The reliability of machine learning, and advice generated by machine learning, is variable.

One way to deal with this is to limit the impact the advice has on the algorithm to a certain extent. There will always be a trade off, more trust in the advice means there is a higher risk, but there is also a higher reward. The online algorithm doesn't have to follow the advice at all times. If certain advice leads to an undesirable outcome the algorithm might not follow advice in this case and instead continue without advice. Making decisions like this impact the competitive ratio as well. More complex algorithms can result in a better competitive ratio, but also a more complex one.

In the buy or rent problem the advice can be whether the player should

buy on the first day. Following this advice no matter what can lead to a bad outcome. If the advice is correct then the online algorithm will be as good as the optimal solution. If the advice is incorrect depending on how far off the advice is the cost of the online algorithm with advice can be much higher than an algorithm without advice. A safer option, for example, would to buy if the advice tells the algorithm to buy and otherwise use an online algorithm without advice to determine whether to buy or not. This enables the algorithm to benefit from correct predictions while still limiting the competitive ratio of the algorithm when the advice is incorrect.

5 Advice for one-way trading

Two types of advice were tested as an addition for the one-way trading problem. The first is a prediction for the day on which the highest exchange rate will be presented, the second is what the highest presented rate will be. Both of these types of advice are a single integer that is calculated at the start of the one-way trading game.

The algorithm that uses advice will set aside a part of the budget that is saved for the predicted event. The rest of the budget will be spend according to the standard online algorithm for one-way trading. The amount of budget that is set aside depends on the trust that is put in the prediction.

5.1 The maximum rate

If the advice is correct then the online algorithm can achieve a competitive ratio of 1 by simply waiting for the predicted ratio and spending the budget at this rate.

With this type of advice there are two possible scenarios, the prediction can be too high or it can be too low. These two cases are very different. When the advice is too low the algorithm will still spend the saved budget but this will be done at a rate less than OPT. If the prediction is too high then the algorithm will never spend and will hold onto the savings until the last day. On the last day it will be forced to spend the savings on the current rate.

The prediction error ϵ is calculated through the difference in the predicted rate and the actual maximum rate that was presented to the player.

An advantage of this prediction is that even if ϵ is small the ratio between the ALG and OPT can still be low. This is only true if the error is below the maximum rate. The downside being that if the prediction is too high then the predicted rate will never come and the algorithm will only spend the saved budget on the last day. This can result in a very low profit, when the last rate is equal to m for example.

5.2 The day with the maximum rate

If the advice is correct then the online algorithm can achieve a competitive ratio of 1 by simply waiting for the predicted day and then spend the entire budget on this day.

The error ϵ is calculated by the difference in days between the prediction and the correct day. An error of one means that either the next or the previous day would be the correct prediction. Since the player knows n, the advantages of this advice is that the predicted day will always occur. Even if the predicted day might not be correct the algorithm will spend the saved part of its budget.

The disadvantage is that even with a very low error the competitive ratio can be very bad. A difference in rates when there is one day between the prediction and the true value can be large. When using data from the real world a difference of one will almost never be very large but there are still exceptions to this.

6 Two competitive ratios

Instead of using a single competitive ratio, two competitive ratios can be used to measure the impact advice has on online algorithms [1].

Any online algorithm ALG that uses advice has the competitive ratio r_{ALG} which is the competitive ratio when the advice is correct and the competitive ratio w_{ALG} which is the competitive ratio if the advice is incorrect. w_{ALG} can not be better than the competitive ratio of the best known online algorithm without advice. The best case for w_{ALG} is to ignore the advice, which would lead to the same competitive ratio as an online algorithm that does not use advice. r_{ALG} will be at least as good as w_{ALG} since the advice is assumed to be correct.

These competitive ratios should not only take all possible input instances into account, but also all possible instances of advice. r_{ALG} represents the competitive ratio of algorithm ALG on all possible input instances σ using advice ϕ where ϕ is the perfect advice. w_{ALG} represents the competitive ratio on all possible input instances σ using advice ϕ where ϕ is the worst possible advice. For w_{ALG} we essentially assume that a malicious entity is giving advice to the algorithm, following this advice will result in the worst possible performance.

$$r_{\text{ALG}} = \sup_{\sigma} \inf_{\phi} \frac{\text{ALG}(\sigma, \phi)}{\text{OPT}(\sigma)} \text{ and } w_{\text{ALG}} = \sup_{\sigma} \sup_{\phi} \frac{\text{ALG}(\sigma, \phi)}{\text{OPT}(\sigma)}$$

Using r_{ALG} and w_{ALG} an online algorithm with advice has a tuple of competitive ratios $(r_{\text{ALG}}, w_{\text{ALG}})$. To compare different algorithms the notion of dominance is used. Algorithm A **dominates** algorithm B if $r_A \leq r_B$ and $w_A < w_B$ or $r_A < r_B$ and $w_A \leq w_B$. When comparing a (4,8)-competitive algorithm it's easy to see this is better than a (4,10)-competitive algorithm. This isn't true when comparing it to a (6,4)-competitive algorithm.

6.1 The competitive ratio for one-way trading with advice

To calculate the competitive ratios r_{ALG} and w_{ALG} a number of extra values are also needed. p^* is the true maximum rate so the correct prediction and p_{pred}, the actual prediction. The budget b, the trust λ and the achievable competitive ratio c.

The trust in the advice λ will be used to save a certain part of the budget to spend on the prediction, λ has a value between 0 and 1. 0 signifies no trust in the advice while 1 is full trust in the advice. Using a budget b, $\lambda * b$ will be set aside to spend on the maximum predicted rate. The remaining $(1 - \lambda) * b$ will be spent according to the algorithm without advice. When using $\lambda = 0$ the algorithm will be the same as the standard online algorithm without advice.

6.1.1 Advice is the maximum rate

We need to separate the competitive ratio for this advice into three separate ratios. One where the advice is correct and two when the advice is incorrect. If the advice is incorrect the prediction can either predict a rate that is too high, in that case the algorithm will not spend since the prediction never happens. The other case is when the prediction is too low, in this case the algorithm will spend at a rate that is below the actual maximum rate.

$$\frac{\text{OPT}(I)}{\text{ALG}(I)} \leq c$$
$$\text{ALG}(I) \geq \frac{\text{OPT}(I)}{c}$$

When the advice is correct, r_{ALG}.

$$\begin{aligned}
\text{OPT} &= b * p^* \\
\text{ALG} &\geq \frac{(1-\lambda) * b * p^*}{c} + \lambda * b * p^* \\
\frac{\text{OPT}}{\text{ALG}} &\geq \frac{b * p^*}{\frac{(1-\lambda)*b*p^*}{c} + \lambda * b * p^*} \\
&\geq \frac{1}{\frac{(1-\lambda)}{c} + \lambda} \\
&\geq \frac{c}{(1-\lambda) + \lambda * c}
\end{aligned}$$

The advice is incorrect and the predicted rate p_{pred} is lower than the maximum rate p^* so the $\lambda * b$ part of the budget will be spend at a rate p with

$p \geq p_{pred}$.

$$\begin{aligned}
\text{OPT} &= b * p^* \\
\text{ALG} &\geq \frac{(1-\lambda) * b * p^*}{c} + \lambda * b * p_{pred} \\
\frac{\text{OPT}}{\text{ALG}} &\geq \frac{b * p^*}{\frac{(1-\lambda)*b*p^*}{c} + \lambda * b * p_{pred}} \\
&\geq \frac{p^* c}{(1-\lambda) * p^* + \lambda * p_{pred} * c}
\end{aligned}$$

The advice is incorrect and the predicted rate p_{pred} is higher than the maximum rate p^* so the $\lambda * b$ part of the budget will be spent on the last day. The only thing known about the last rate p is that $p \geq m$.

$$\begin{aligned}
\text{OPT} &= b * p^* \\
\text{ALG} &\geq \frac{(1-\lambda) * b * p^*}{c} + \lambda * b * m \\
\frac{\text{OPT}}{\text{ALG}} &\geq \frac{b * p^*}{\frac{(1-\lambda)*b*p^*}{c} + \lambda * b * m} \\
&\geq \frac{p^* c}{(1-\lambda) * p^* + \lambda * m * c}
\end{aligned}$$

Since for all rates $p \geq m$ this means that $p_{pred} \geq m$. From this it's easy to see that a prediction that is higher than p^* is always equal to or worse than a prediction that is lower than p^*. This means that w_{ALG} is when the predicted rate is too high.

6.1.2 The day

There are only the two possible cases when using advice that predicts the day. Either the advice is correct or incorrect. Any incorrect prediction can result in the algorithm spending the saved budget on a day with exchange rate m in the worst case.

The advice is correct r_{ALG}.

$$\begin{aligned}
\text{OPT} &= b * p^* \\
\text{ALG} &\geq \frac{(1-\lambda) * b * p^*}{c} + \lambda * b * p^* \\
\frac{\text{OPT}}{\text{ALG}} &\geq \frac{b * p^*}{\frac{(1-\lambda)*b*p^*}{c} + \lambda * b * p^*} \\
&\geq \frac{1}{\frac{(1-\lambda)}{c} + \lambda} \\
&\geq \frac{c}{(1-\lambda) + \lambda * c}
\end{aligned}$$

The advice is incorrect w_{ALG}. When the advice is incorrect then the exchange rate on the day can be any possible rate. The worst rate will be the minimum rate.

$$\begin{aligned}
\text{OPT} &= b * p^* \\
\text{ALG} &\geq \frac{(1-\lambda) * b * p^*}{c} + \lambda * b * m \\
\frac{\text{OPT}}{\text{ALG}} &\geq \frac{b * p^*}{\frac{(1-\lambda)*b*p^*}{c} + \lambda * b * m} \\
&\geq \frac{p^* c}{(1-\lambda) * p^* + \lambda * m * c}
\end{aligned}$$

6.2 Error

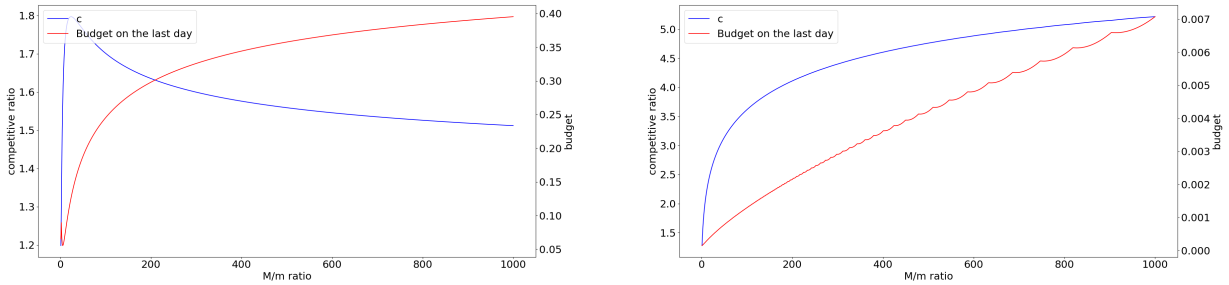
The prediction error can be calculated as the absolute difference between p^* and the predicted rate, $|p^* - p_{pred}|$.

Both the maximum prediction and the day prediction have the same r_{ALG} and w_{ALG} (Equation 3). This means that neither of them dominates the other.

$$(r_{\text{ALG}}, w_{\text{ALG}}) = \left(\frac{c}{(1-\lambda) + \lambda * c}, \frac{p^* c}{(1-\lambda) * p^* + \lambda * m * c} \right) \tag{3}$$

7 Experiments

7.1 Setting the benchmarks



(a) $n = 5$ with increasing M/m

(b) $n = 2500$ with increasing M/m

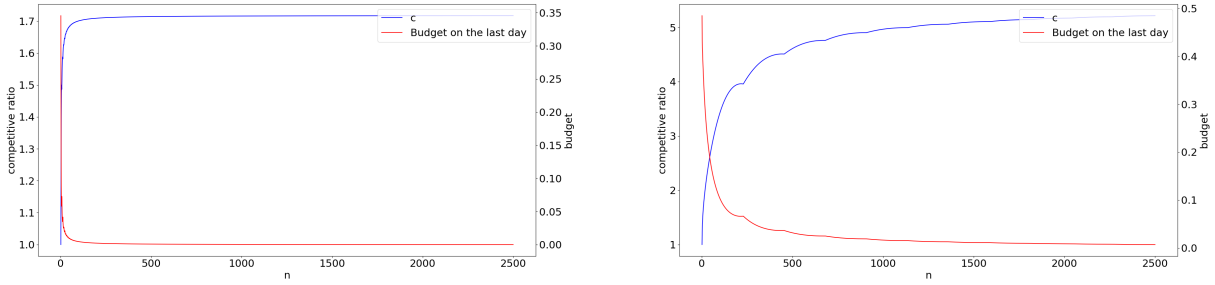
Figure 2: The influence of the remaining budget on the last day on c.

In the experiments either n will be fixed while M/m goes up or M/m is fixed while n goes up to show the exact effects these values have on the online algorithm. When either of the values increases the competitive ratio should

also increase. m is fixed during the experiments and only M will be changed to increase M/m. With an adversarial opponent the difference between p_i and p_{i-1} is constant. The experimental results don't use an very large M/m or n, which is assumed in the threat based approach. This influences experiments done with smaller values because the step size between two rates, $(p_i - p_{i-1})$, will be bigger with a smaller n. The opposite is true when decreasing M/m, $(p_i - p_{i-1})$ will get smaller. This can also be seen when comparing Figure 2 with Figure 3. When increasing M/m the remaining budget will increase while increasing the n will cause the remaining budget to decrease.

When using smaller values for n this results in the algorithm keeping a bigger part of the budget for the last day. With an adversarial opponent the last day will have the best rate and so the algorithm will earn a higher profit by saving (Figure 2a). This becomes less of a problem as n increases, but will make the experimental results deviate from the theoretical results if the number of presented rates is too small, a large enough value for n is needed (Figure 2b).

When the number of days goes up the amount that is saved for the last day will be less and less significant and the experimental results will almost match the theoretical ones. With a very large number of days, as was assumed when applying the algorithm in theory, the remaining budget on the last day is not significant. This means that a large number of days is needed for reliable experimental results. $n = 2500$ is large enough for the remaining budget on the last day to be negligible. When using a very high M/m even $n = 2500$ still has a visible effect but for the experiments smaller values will be used for M/m.



(a) $M/m = 5$ with increasing n

(b) $M/m = 2500$ with increasing n

Figure 3: The influence of the remaining budget on the last day on c.

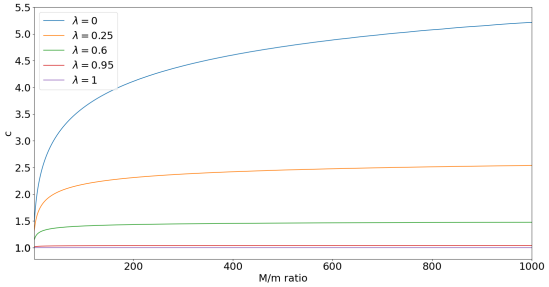
Increasing M/m does not have the same effect on the performance. Increasing M/m will result in a larger value for $p_i - p_{i-1}$ as well as a larger value for c (Figure 3). This means that results will be similar for different values for M/m. As can be seen in the figures, a higher value for M/m means that the competitive ratio will be higher. There is a difference in how many days are needed for the online algorithm to achieve close the limit for the competitive ratio as well.

7.2 Experiments with advice

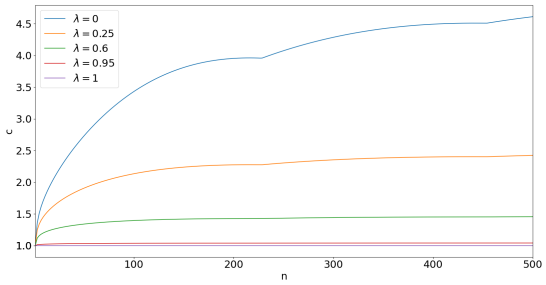
Advice has a λ between 0 and 1, for each experiment multiple levels of trust are tested. A trust of 0 is also used to show how well the algorithm without advice performs compared to algorithms with advice on the same input sequences. First the experiments with perfect advice will be compared to each other. After looking at the cases with perfect advice the two different kinds of advice are split into their own separate sections.

7.2.1 Experiments with Perfect advice

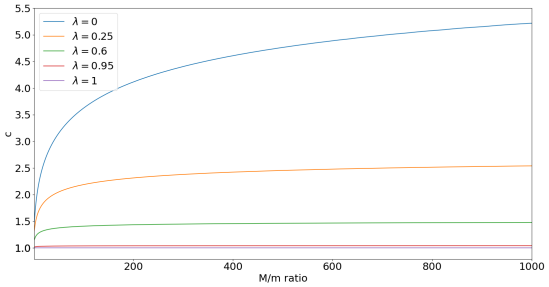
These experiments are done with advice that enables the algorithm to achieve a competitive ratio of 1. For the maximum advice this means that the advice is the highest maximum rate that will be presented to the player. M only shows the maximum possible rate, but this rate might not be presented to the player. For the advice on the day to trade on the advice will be the day with the highest rate. In both cases the online algorithm will trade on the same day as the optimal solution would.



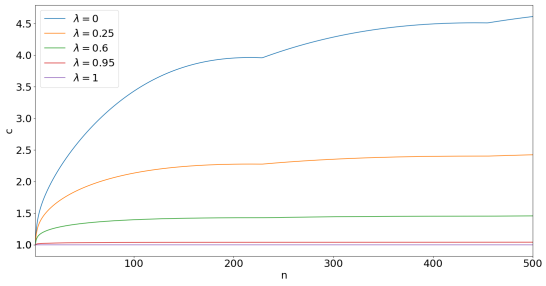
(a) Advice is the maximum rate, n is fixed at 2500 and M/m is increasing till 1000.



(b) Advice is the maximum rate, M/m is fixed at 1000 and n is increasing till 500



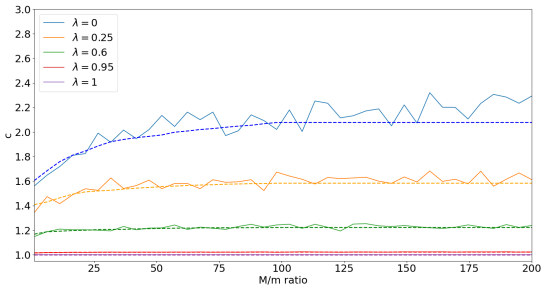
(c) Advice is the day with the highest rate, n is fixed at 2500 and M/m is increasing till 1000



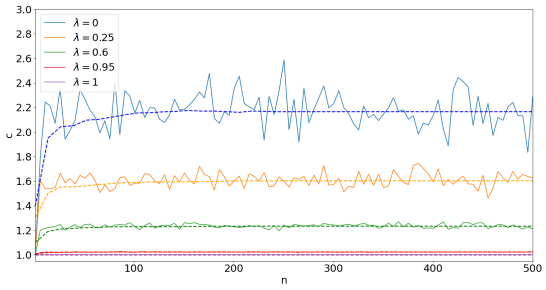
(d) Advice is the day with the highest rate, M/m is fixed at 1000 and n is increasing till 500

Figure 4: Experiments done with perfect advice on adversarial rates.

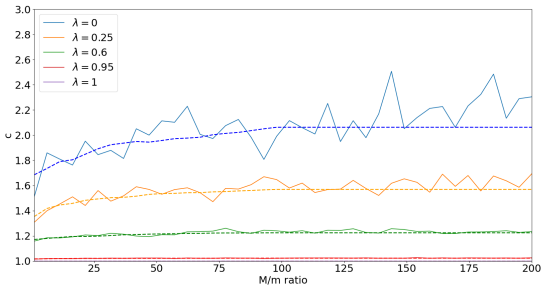
It's easy to see that both types of advice perform the same when using adversarial rates (Figure 4). When the advice is perfect and the algorithm has complete trust a competitive ratio of 1 is achievable. As soon as this is no longer the case the algorithm will have a higher competitive ratio, and a worse performance.



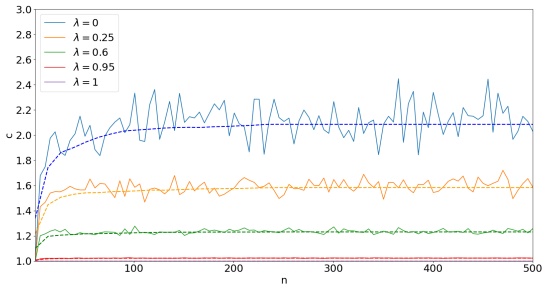
(a) Advice is the maximum rate, n is fixed at 2500 and M/m is increasing till 200.



(b) Advice is the maximum rate, M/m is fixed at 200 and n is increasing till 500.



(c) Advice is the day, n is fixed at 2500 and M/m increasing till 200.



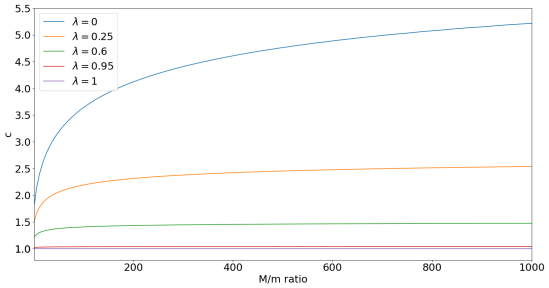
(d) Advice is the day, M/m is fixed at 200 and n is increasing till 500.

Figure 5: Experiments done with perfect advice on uniform random rates, the algorithms were repeated 100 times.

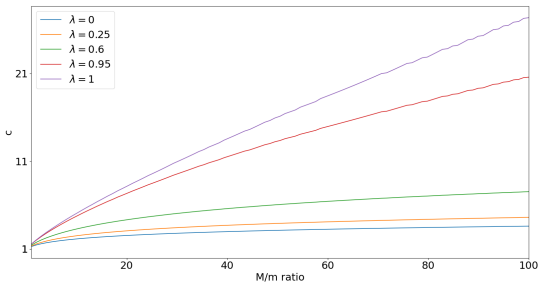
When using random rates and perfect advice the algorithm with advice is still clearly better than the algorithm without advice (Figure 5). The dotted line shows the average competitive ratio c from all smaller values including the current one. This average value has been smoothed to show results more clearly. Even though n was only increased till 500 in the case of random rates the average competitive ratio stabilises earlier than this. Much earlier than the 2500 that was needed when using adversarial rates. In all of the experiments the competitive ratio's are limited at about the same values for each respective level of trust.

7.2.2 Advice is the maximum rate

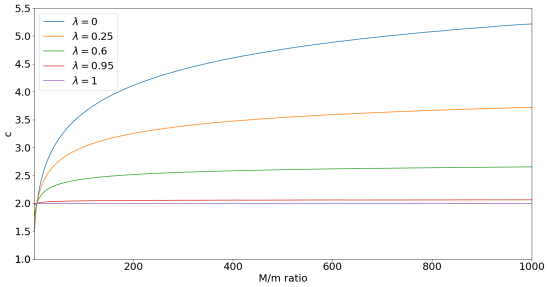
First advice that predicts the maximum rate that will be presented to the algorithm will be discussed. Like with the perfect advice the first experiments will be done with adversarial rates and then random rates will be used.



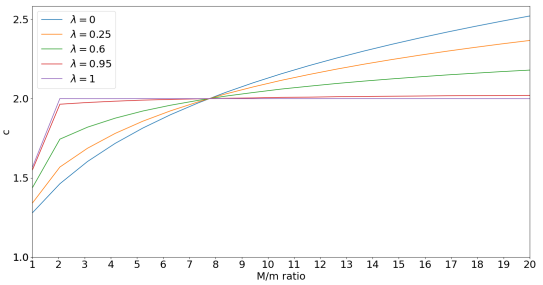
(a) Results when the advice is higher than the optimal advice = OPT*1.5



(b) Results when the advice is lower than the optimal advice = m.



(c) Results when the advice is lower than the optimal advice = OPT/2

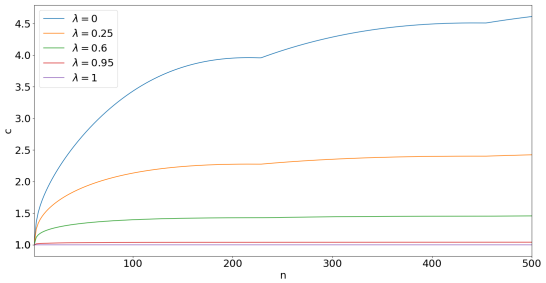


(d) Results when the advice is lower than the optimal advice = OPT/2, $M = 20$

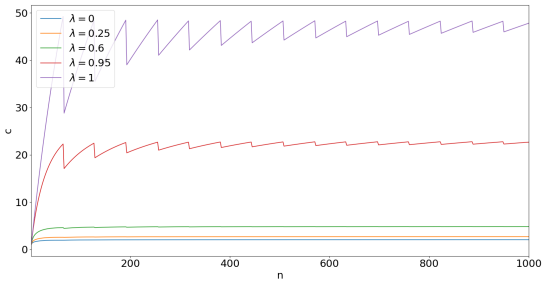
Figure 6: Experiments done with $n = 2500$ with increasing M/m.

Adversarial rates with a fixed n and increasing M/m Both advice that is too high and advice that is too low results in almost equal performance given enough rates (Figure 6a, 6c). The main difference is when the ratio between M and m is low. Figure 6d shows that just after a ratio of eight the algorithm that uses advice starts performing better than the algorithm without advice. When the advice is too low the algorithm will end up spending the budget at a bad rate, since the advice depends on the ratio between M and m in this case there is a certain point where the advice becomes good enough so the algorithm with advice outperforms. When the prediction is too high something different happens. In this case the budget is saved until the last day. Since these experiments are done with adversarial rates the longer the budget is saved the

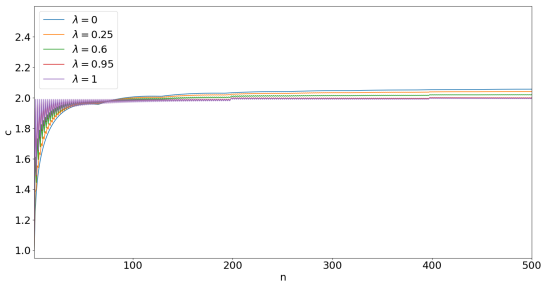
better the algorithm will perform. When the advice is the lowest possible rate the algorithm that uses this advice will simply spend the budget on the first day. Since the rates are only going up for this experiment this is the absolute worst choice. The effects of this choice are a linearly increasing competitive ratio (Figure 6b).



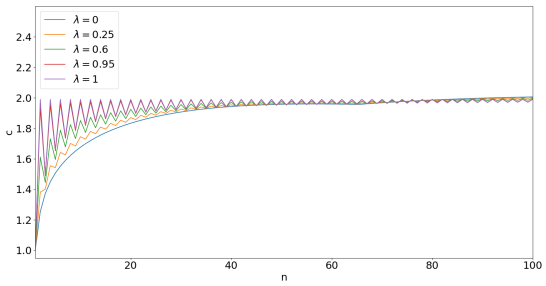
(a) Results when the advice is higher than the optimal advice = OPT*1.5



(b) Results when the advice is lower than the optimal advice = m.



(c) Results when the advice is lower than the optimal advice = OPT/2



(d) Results when the advice is lower than the optimal advice = OPT/2, $n = 100$

Figure 7: Experiments done with $m/M = 200$ with increasing n.

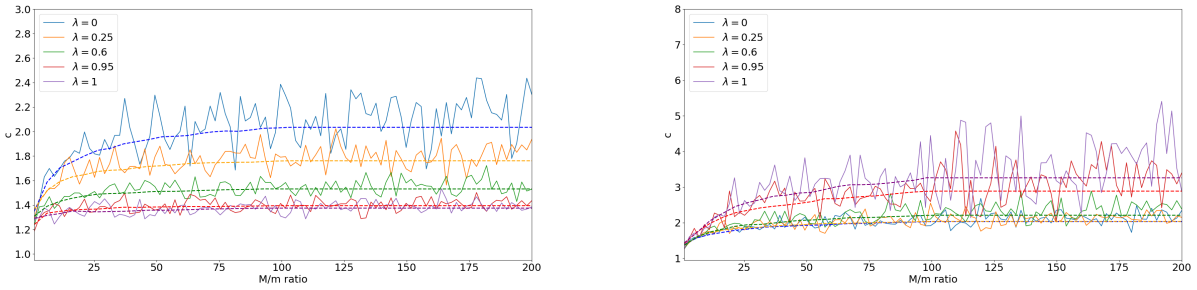
Adversarial rates with a fixed M/m and increasing n When using adversarial rates an advice that is too high will have the same results as perfect advice would have. Since the predicted maximum rate never is presented the algorithm will spend on the last day. With adversarial rates this day will be the optimal rate (Figure 7a).

The competitive ratio with a low n is heavily dependant on the remaining budget on the last trading opportunity. There are certain points at which the competitive ratio goes down even though it should increase as n increases. When the advice is too low, but not the minimum (Figure 7c, 7d) the performance of the algorithm fluctuates up and down, the variance goes down as n goes up. This is examined further in Section 7.3. There is a certain point where the

algorithm with advice starts outperforming the algorithm without advice. This happens after 88 rates.

The fluctuation is caused by the fact that the algorithm only trades above a certain minimum rate depending on c. There are breakpoints where the first rate is not high enough for the algorithm to consider trading which causes more of the budget to be saved. Since both the value of c and the step size between rates relies on n this causes the competitive ratio to fluctuate.

Rates drawn from a uniform random distribution. These experiments were done by generating the rates from a uniform random distribution multiple times. The average was then taken of all of the results, in these experiments the rates were generated 100 times. Each graph has a trend line that shows the average competitive ratio of all the smaller values of the x-axis, M/m or n respectively. The line has been smoothed using the average value of all competitive ratios for that line.

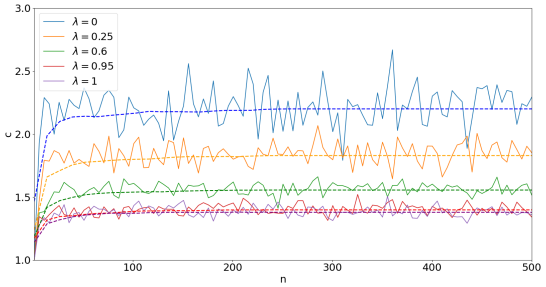


(a) The advice is lower than the optimal advice = OPT/2. (b) The advice is lower than the optimal, advice = m.

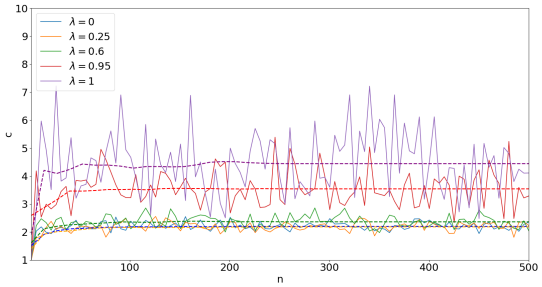
Figure 8: Experiments done with $n = 2500$ with increasing M/m, M/m goes up to 200. The rates are drawn from a uniform random distribution.

Uniform random rates with a fixed n and increasing M/m. With an increasing M/m the error of the advice still has influence on the competitive ratio even with random rates. There is a clear difference in the average competitive ratios when comparing the different errors (Figure 8). As was the case before, with an advice of m the algorithm without advice performs better than the algorithm with advice. The trend of the competitive is upwards as M/m increases.

Uniform random rates with a fixed M/m and increasing n. Even though the random generation is the same as with a changing M/m the graphs look different. With an advice of m the algorithm without advice performs better. The increasing of the n makes the variance of the graph increase compared to increasing the M/m (Figure 9).



(a) The advice is lower than the optimal, advice = OPT/2.



(b) The advice is lower than the optimal, advice = m.

Figure 9: Experiments done with $M/m = 1000$ with increasing n, n goes up to 500. The rates are drawn from a uniform random distribution.

There is still a clear difference between the different levels of trust. The difference in increasing the value of n is also noticeable.

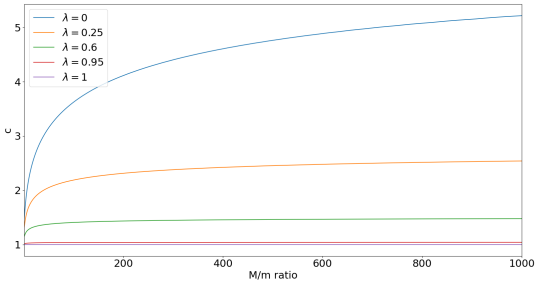
Depending on the error of the advice the fluctuation is also different. A lower error results in more stable results, a higher error has much more unstable results, this can be seen in both Figure 8b and Figure 9b.

7.2.3 Advice is the day to spend

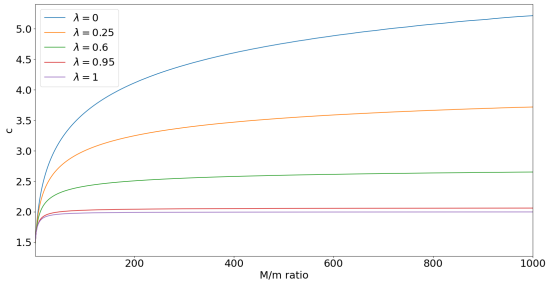
Experiments done when the advice is which day to spend the budget on. Different error values were used for the experiments. The error is the difference in days between the predicted value and the correct value. Two different values for the error were tested. An error of one was used, meaning the prediction is one day off from the correct prediction and an error of $n/2$ was used, this would be a significant error. A worst case prediction was also tested, in this case there is not a fixed error but the error depends on the difference in the number of days that would be between the day with the highest rate and the day that is the lowest rate.

Adversarial rates with a fixed n and increasing M/m When the error is one the algorithm performs close to when the advice is the optimal rate (Figure 10a). When using adversarial rates if the error is low the difference between a small number of days will be low. When the error is $n/2$ the algorithm with advice still performs much better than the algorithm without advice (Figure 10b). When the advice is correct within a certain margin the algorithm is improved by advice, even being off by $n/2$ days will still improve the performance. There is still a point where the advice will cause the algorithm that uses advice to perform worse than the algorithm without advice (Figure 10c).

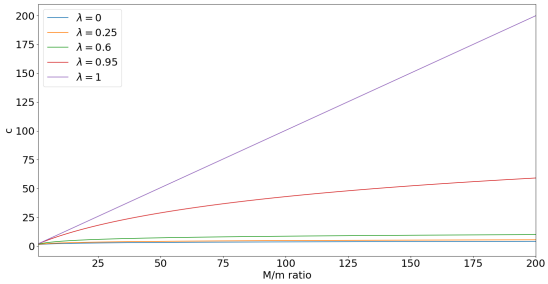
Adversarial rates with a fixed M/m and increasing n When using the adversarial rate and a low number of days the difference between days is very



(a) The advice is wrong by a single rate.



(b) The advice is wrong by $n/2$ rates.

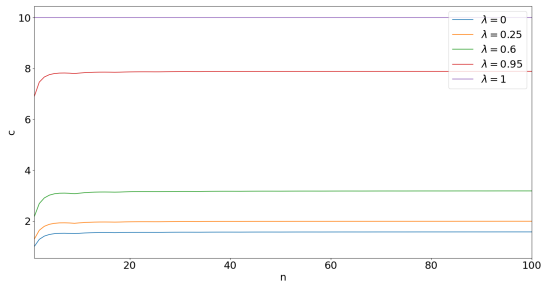


(c) The advice is the day with the lowest rate, M/m goes to 200

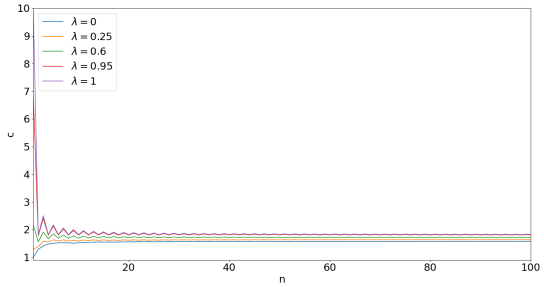
Figure 10: Experiments done with fixed $n = 2500$ with increasing M/m, increasing to 1000.

significant (Figure 11). With the worst advice the algorithm without advice performs better (Figure 11a). With a significant error the algorithm without advice still outperforms the algorithm with advice (Figure 11b). There is a certain point where if the advice is good enough the algorithm with advice will outperform, when given enough rates to trade on (Figure 11d). Even though increasing the number of trading opportunities should increase the competitive ratio, a certain amount of opportunities are needed for the algorithm with advice.

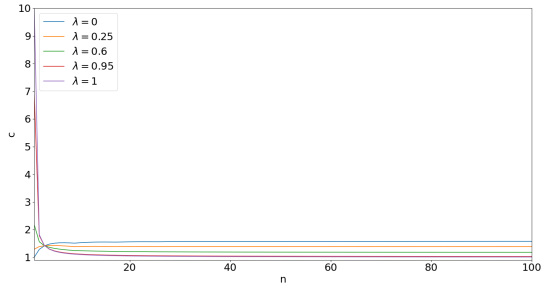
Rates drawn from a uniform random distribution. These experiments were done by generating the rates from a uniform random distribution multiple times. As with the maximum rate advice the average was then taken of all of the results, in these experiments the rates were generated 100 times. Each graph has a trend line that shows the average competitive ratio of all the smaller values of the x-axis, M/m or n respectively. The line has been smoothed using the average value of all competitive ratios for that line.



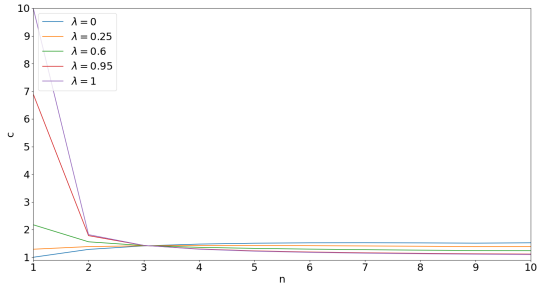
(a) Results when the advice is the day with the lowest rate.



(b) Results when the advice is off by $n/2$ rates.



(c) Results when the advice is off by a single rate.

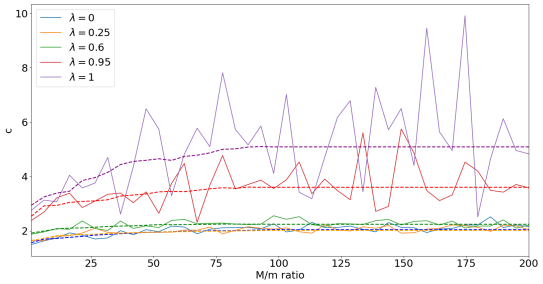


(d) Results when the advice is off by a single rate, the first 10 rates.

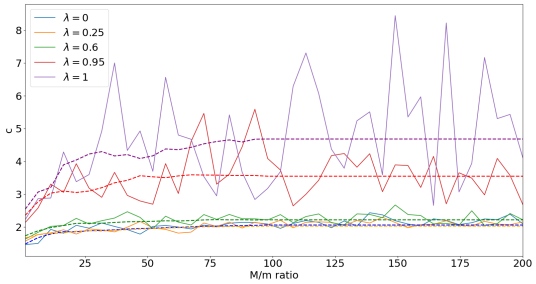
Figure 11: Experiments done with $M/m = 10$ with increasing n, increasing to 100.

Uniform random rates with a fixed n and increasing M/m. When using random rates the algorithm without advice seems to always outperform the algorithm with advice (Figure 12). In later experiments more realistic data is used, but with random rates an error of one can have a huge effect on the competitive ratio. This means that even if the error is one, the algorithm with advice can perform very badly.

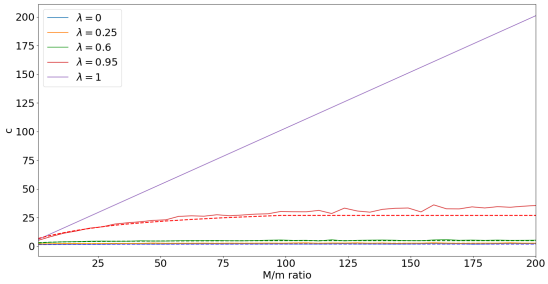
Uniform random rates with a fixed M/m and increasing n. As was the case in Figure 12, even a very low error on the prediction results in the algorithm without advice performing worse than the algorithm with advice. Better advice does have an effect on the performance of the algorithm with advice though. Better advice clearly makes the algorithm perform better (Figure 13a).



(a) Results when the advice is off by one day.

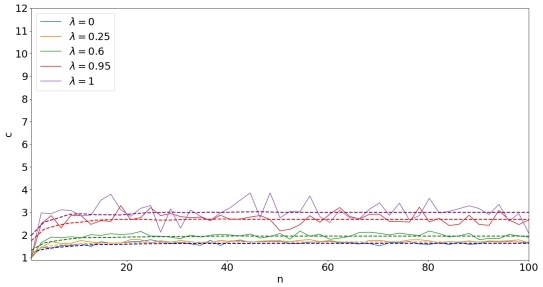


(b) Results when the advice is by $n/2$ days.

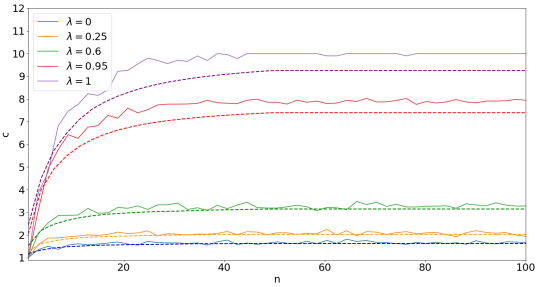


(c) Results when the advice is the day with the lowest rate.

Figure 12: Experiments done with $n = 2500$ with increasing M/m, M/m goes up to 200. The rates are drawn from a uniform random distribution.



(a) Results when the advice is off by one day.



(b) Results when the advice is the day with the lowest rate.

Figure 13: Experiments done with $M/m = 10$ with increasing n, n goes up to 100. The rates are drawn from a uniform random distribution

7.3 Fluctuation in the Competitive Ratio

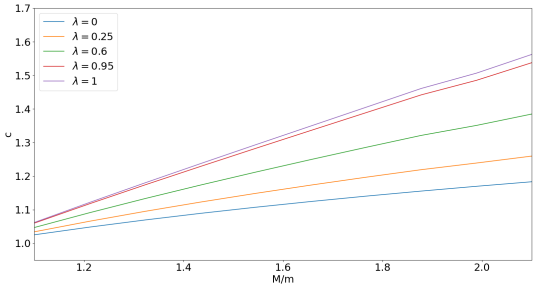
Figure 7b shows a very high fluctuation even though you would expect the competitive ratio to go up as n increases. Since adversarial rates are used this means that OPT always trades at a rate of M, which is presented on the last possible trading opportunity. This leaves a fluctuation of the performance of ALG.

The fluctuation happens because of the rule that the algorithm will only spend on rates above $c*m$. With adversarial rates the step size between rates is always equal, there are certain breakpoints where $c <$ this step size. This means that the for the first rate p_1, $p_1 \geq c * m$ so the algorithm will spend. There are certain values for n and m where $p_1 < c * m$ so the algorithm will not spend. Instead it will spend on p_2, which has a rate that is twice as high as p_1. The next break point will happen when $p_2 \geq c * m$ after which the algorithm will spend on p_3. This difference is less, $p_3 = p_2 * 2/3$, so the difference in the two succeeding competitive ratio's is also smaller. The effects of this can be best seen in Figure 7b

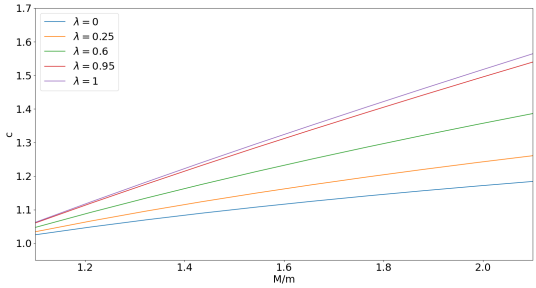
7.4 Experiments with a low M.

When using the maximum rate as the advice on adversarial rates a M/m of over 8 was needed before the algorithm with advice outperformed the algorithm without advice. Even though a ratio of 8 is possible it will be rare in the real world. There are cases where it happens if a big enough time period is taken. Taking the apple stock for example, going back 5 years results in $M/m = 182.94/36.43 \approx 5.0$. When the time scale is smaller a more realistic assumption is that $M/m < 2$. When looking at something more stable, like the Dollar and Euro, a $M/m < 2$ is expected. Even if you take the last 13.5 years M/m barely goes above 1.5. With low values for M and m the value of c will be very close to 1.

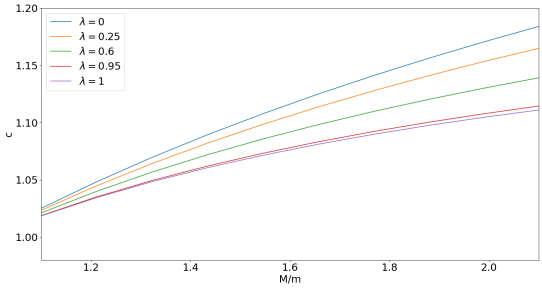
When using a small value for M/m the difference in n becomes a lot less noticeable. There is a difference between $n = 100$ and $n = 2500$ but this difference in competitive ratio is small (Figure 14a, 14b). As with the previous experiments an advice that is wrong by OPT/2 has an error that is large enough for the algorithm without advice to outperform the algorithm with advice. When the error of the advice is reduced, the algorithm with advice performs better even at a low M/m (Figure 14c)



(a) n=100 and advice is lower than the optimal advice = OPT/2



(b) n=2500 and advice is lower than the optimal advice = OPT/2



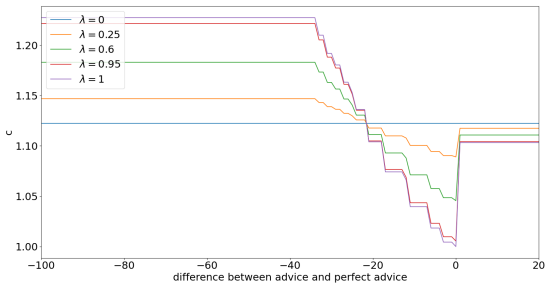
(c) n=2500 and advice is lower than the optimal advice = OPT $* \frac{4}{5}$

Figure 14: Experiments done with a fixed n and increasing M/m, M/m goes up to 2. Adversarial rates are used.

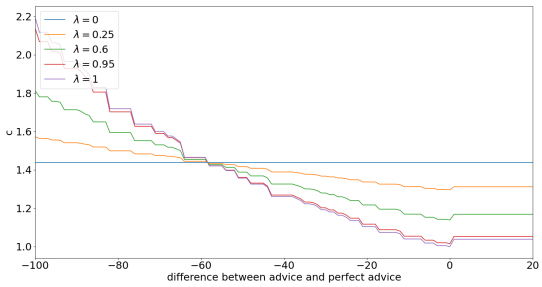
Both the advice that predicts the higest rate and the best day to trade on were applied to historical Apple stock value and historical Euro to Dollar exchange rate data. Figures 15a and 16a were generated from the historical data from the past 10 months. Figures 15b and 16b were generated with data from the past five years. The data was collected from https://www.nasdaq.com/. The values for m and M are in Dollars.

When the advice gives the maximum rate an error that is too high results in a better performance compared to the algorithm without advice. This is only because the most recent stock price has been relatively high. Especially when compared to how low valued the stock was 5 years ago. The best performance is still with an error of 0. Even if the error is high, if one of the first rates presented is a relatively high rate an increase in the error won't matter (Figure 15a).

The results when the advice is the day to trade are very different depending on the time scale. With a large n a error of 100 will be less significant than with a small n (Figure 16). $n = 1258$ in Figure 3b, so when the error is within

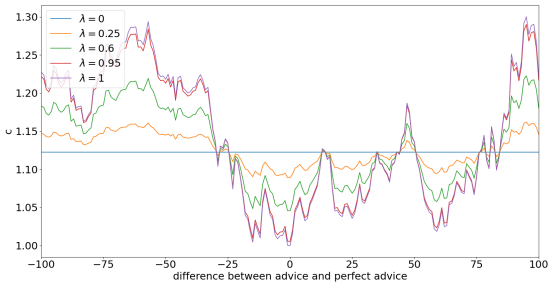


(a) Error from -100 to +20, $M/m \approx 1.38$. $m = 132.39$ and $M = 182.94$

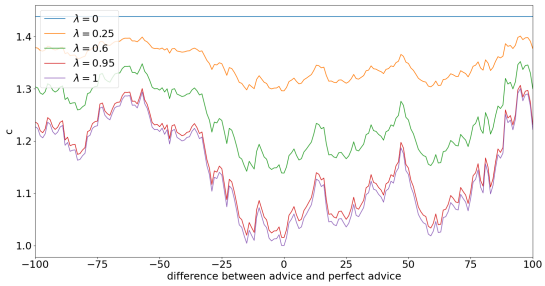


(b) Error from -100 to +20 $M/m \approx 5.02$. $m = 36.43$ and $M = 182.94$

Figure 15: The online algorithm using the maximum rate as advice applied on historical Apple stock value. The error was changed from -100 to +20.



(a) Error from -100 to +100, $M/m \approx 1.38$
$m = 132.39$ and $M = 182.94$



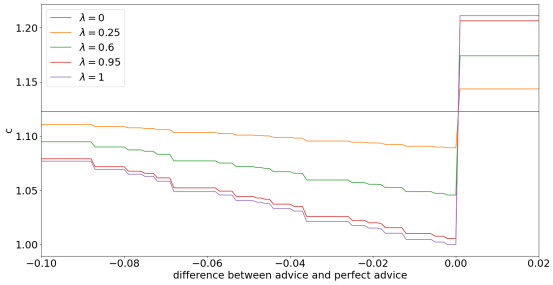
(b) Error from -100 to +100, $M/m \approx 5.02$
$m = 36.43$ and $M = 182.94$

Figure 16: The online algorithm using the day as advice applied on historical Apple stock value. The error was changed from -100 to +100.

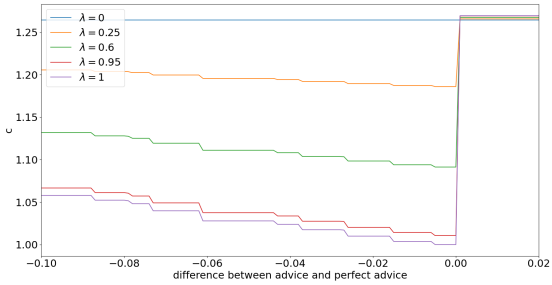
a margin of almost 16% off from the perfect prediction the performance is still better in all cases.

The algorithm with both kinds of advice was also applied to a much more stable currency, the Euro. The Euro will be compared to the Dollar. Figures 17a and 18a were generated from the historical data from the past 2 years. Figures 17b and 18b were generated with data from the past 13.5 years. The data was fetched from https://www.investing.com/. The values for m and M are in Dollars.

The M/m is much lower than with the previous example, the competitive ratio's are also lower. When the advice predicts the maximum rate the algorithm is punished for overestimating. Since you can get less Dollars for Euros that you could in the past the algorithm gains a smaller profit by saving. When

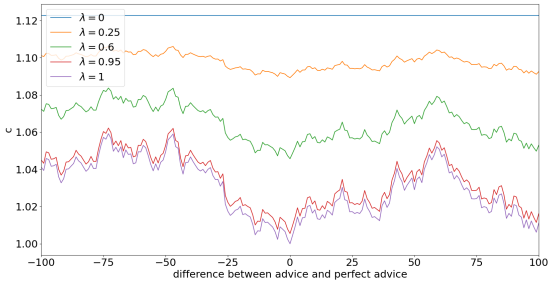


(a) $M/m \approx 1.230$.
$m = 1.0018$ and $M = 1.2327$

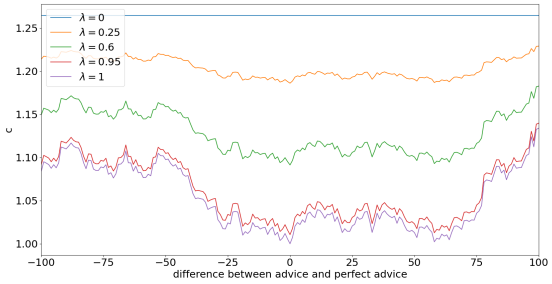


(b) $M/m \approx 1.529$.
$m = 0.8268$ and $M = 1.2641$

Figure 17: The online algorithm using the maximum rate as advice applied on historical Euro to Dollar exchange rates. The error was changed from -0.1 to +0.02.



(a) $M/m \approx 1.230$.
$m = 1.0018$ and $M = 1.2327$



(b) $M/m \approx 1.529$.
$m = 0.8268$ and $M = 1.2641$

Figure 18: The online algorithm using the day as advice applied on historical Euro to Dollar exchange rates. The error was changed from -100 to +100.

the advice predicts a maximum that is too low then the performance if better. Even when the error is pretty significant. $M/m = 1.230$ so even with an error of 42% the algorithm with advice outperforms the algorithm without advice, so long as the error is an underestimation. When the advice predicts the day the performance is always better in this case. n is large in both cases here, $n = 688$ for Figure 18a and $n = 5000$ for Figure 18b.

8 Conclusion

The biggest difference between the theoretical and practical results comes from the fact that the amount of budget left on the last day can impact the results of the algorithm. If there are not enough trading opportunities then the impact of this is significant. n has to be large, at least 2500, for it not to impact the results significantly when using adversarial rates.

Adversarial rates are useful for figuring out the competitive ratio but not as useful when trying to incorporate the advice tested for this paper. If the algorithm tends to save for a specific rate and the best rate is presented at the end then the algorithm will perform better depending on how much is saved for the last rate.

The theoretical results show that depending on n and M/m the quality of the advice needs to be higher. With a low M/m especially the quality of the advice is important. Perfect advice will always result in the algorithm with advice performing at least as good as the algorithm without advice. This means there is always a point where the advice is good enough for the online algorithm with advice to outperform the algorithm without advice.

When data from Apple and the exchange rate from Dollars to Euro's was used this was no longer the case. Even with a low M/m and a high error in the advice the algorithm that used the prediction for the best day still outperformed the online algorithm on Dollar to Euro exchange rate data. This has to do with the fact that the exchange rates are much more stable than is the case with adversarial rates and with random rates.

There are too many variables involved to make a definitive conclusion about whether an online algorithm with advice generated by machine learning will be better than online algorithms without advice. The results are promising though, and if the machine learned advice has a low enough error the results will be better than not using advice.

8.1 Two way trading

Two-way trading is one-way trading with one addition. The player can still trade budget, X into profit, Y, but now the player can also trade back. There is a single exchange rate, Y to X uses the inverted exchange rate of X to Y. The goal is to maximise Y, like with the one-way trading problem the remaining amount X is irrelevant. This means the algorithm will trade all remaining X to Y on the last day. This adds another layer of complexity, which raises the maximum Y both the online algorithm and the optimal solution can achieve.

The sequence of exchange rates can be seen as a series of upward- and downward runs. The optimal solution will convert all X to Y at the end of an upward run, and all of the Y to X at the end of a downward run. The one-way trading algorithm can be applied to each individual run. Where the one-way trading will convert X to Y on an upwards run and Y to X on a downwards run.

Expanding one-way trading with advice to two-way trading with advice will need a number of extensions, the budget is assumed to be only decreasing, the

profit is assumed to be only increasing. This is acceptable for one-way trading but not for two-way trading. With two-way trading we can split the sequence of rates into l sets of upwards and downwards runs respectively [4]. As was seen in the experiments the length of downwards and upwards runs can impact results significantly if they are not long enough. Applying this algorithm on real world data might not be effective given the fact that's it's unrealistic to assume long upwards or downwards runs.

When adding advice to two-way trading it might be interesting to split the rates into parts that are long enough and then use prediction to see weather it's more profitable to assume if it will be a downwards run or an upwards run. Since the nature of the algorithm is online the advice can be requested at the start of each sequence of rates and will be in effect be a series of applications of the one-way trading algorithm.

References

[1] Spyros Angelopoulos et al. "Online computation with untrusted advice". In: *arXiv preprint arXiv:1905.05655* (2019).

[2] Thodoris Lykouris and Sergei Vassilvtiskii. "Competitive caching with machine learned advice". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3296–3305.

[3] Manish Purohit, Zoya Svitkina, and Ravi Kumar. "Improving online algorithms via ML predictions". In: *Advances in Neural Information Processing Systems* 31 (2018), pp. 9661–9670.

[4] Ran El-Yaniv et al. "Optimal search and one-way trading online algorithms". In: *Algorithmica* 30.1 (2001), pp. 101–139.