Master's Thesis

Author: Halfdan Rump

Supervisor: Prof. Toshiharu Sugawara

A thesis submitted in fulfilment of the requirements for the degree of

in the

January 2014

XXXX NOT DONE YET XXX

This work proposes a model in which multiple heterogeneous agents use time delayed price information to trade an imaginary financial instrument in a market with a continuous double auction. The main innovation of the model is that, just as in the real world, trading agents do not have access to the market information at the same point in time, which means that the agents generally trade on different information. The model contains agent models of slow human traders using a noisy fundamentalist strategy, and high speed software traders using market maker and chartist strategies. Slow traders base their trading decisions on market information which has been delayed several seconds, while the fast traders observe the market with much smaller delays ranging between a few millisecons to a few hundred milliseconds. By containing agents of such different speeds, the model is relevant to the ongoing discussion of the pros and cons of high frequency trading in financial markets.

The model simulates the market events in the minutes after the advent of bad news represented by sudden negative shock to the fundamental stock price, and does so with a time resolution high enough to register events that unfold from millisecond to millisecond. A genetic algorithm is used to search for model parameters that cause the market to be stable, and parameters that cause the market to crash. Analysis of the results shows that a moderate level of high speed trading activity is not in itself problematic. In fact, high speed market makers are found to reduce price flickering, while high speed chartists are found to decrease the time required for the market to respond to changes in the fundamental price. However, the market is found to respond unfavorably to a large presence of fast traders. First of all, a large presence of fast market makers causes the market to respond sluggishly to the shock, leading to a prolonged disparity between the traded price and the true fundamental price. Secondly, a large presence of fast chartists causes increased flickering of the traded price. Finally it is found that flash-crashes can occur in markets in which ratio of the number of chartists to the number of market makers is high, while at the same time the market makers are faster than the chartists. These results are interesting, as they show both benefits and dangers of having markets where fast computer algorithms trade side by side with human traders.

Acknowledgements

The acknowledgements and the people to thank go here, don't forget to include your project advisor...

Contents

List of Figures

List of Tables

Abbreviations

HFT	**H**igh **F**requency **T**rader
ST	**S**low **T**rader
OB	**O**rder **B**ook
MM	**M**arket **M**aker
SD	**S**tandard **D**eviation
GD	**G**aussian **D**istribution
GMM	**G**aussian **M**ixture **M**odel
GA	**G**enetic **A**lgorithm

Physical Constants

Term	Explanation
ask price	The price of a sell order
bid price	The price of a buy order
model fitnesses	The quantitative measures summarizing how the model behaves
fundamental price	The underlying “true” value of the stock
limit order	Order which is not
market order	
liquidity	The
market depth	
match	When a sell order and a bu order happen to have the same listed price, they are
order book	
order volume	
partial match	Two orders which match, but have different volumes.
share	A fraction of ownership of an asset, such as a stock
spread	
standing order	A market order registered at an order book and waiting for a matching order
tick	The smallest possible price change of the stock
volatility	

Physical Constants

$\mathrm{E_p}[\mathbf{x}]$	The sample mean of vector $\mathbf{x}$
$\mathrm{Var_p}[\mathbf{x}]$	The sampe variance of vector $\mathbf{x}$
$\mathrm{Cov_p}[\mathbf{x}, \mathbf{y}]$	The sample covariance between vector $\mathbf{x}$ and $\mathbf{y}$
$\mathrm{M_p}[\mathbf{x}]$	The sample median of vector $\mathbf{x}$
p^m	Time delay in rounds from agent i to market j. Note that $\tau_{i,j} = \tau_{j,i}$.
s_t	Spread at the end of round t
p_t^a	The lowest askprice in the order book at the end of round t
p_t^b	The highest bidprice in the order book at the end of round t
p^m	Match price, i.e., the price at which a trade is executed
f_t	Fundamental price at round t
p_{fas}	Fundamental price after shock
m_{stable}	Size in ticks of the stability margin around each side of p_{fas}
ρ_{A}	Ratio between the number of chartists and the number of market makers: $\rho_{\mathrm{A}} = \frac{N_c}{N_m}$
ρ_λ	Ratio between the latency of chartists and the latency of market makers: $\rho_\lambda = \frac{\lambda_{c,\mu}}{\lambda_{m,\mu}}$

Symbols

Symbol	Description	Unit
$\texttt{N}_\texttt{r}$	Number of simulation rounds	
$***\lambda$	Average number of ST orders per round	
N_c	Number of HFT SC agents	
N_m	Number of HFT MM agents	
$\lambda_{c,\mu}$, $\lambda_{c,\sigma}$	Mean and SD parameters of the GD for HFT SC latency	
λ_i^c	Latency of HFT SC agent i where $\lambda_i^c \sim \mathcal{N}(\lambda_{c,\mu}, \lambda_{c,\sigma})$	
$T_{c,\mu}$, $T_{c,\sigma}$	Mean and SD parameters of the GD for HFT SC thinking time	
T_i^c	Thinking time of HFT SC agent i where $T_i^c \sim \mathcal{N}(T_{c,\mu}, T_{c,\sigma})$	
$H_{c,\mu}$, $H_{c,\sigma}$	Mean and SD parameters of the GD for HFT SC time horizon	
$H_{c,\sigma}$	Standard deviation of the Gaussian distribution for HFT SC time horizon	
$W_{c,\mu}$, $W_{c,\sigma}$	Mean and SD of the GD for the HFT chartist wait time	
$\lambda_{m,\mu}$	Mean and SD of the GD for HFT MM latency	
λ_m^i	Latency of HFT MM agent i where $\lambda_m^i \sim \mathcal{N}(\lambda_{m,\mu}, \lambda_{m,\sigma})$	
$T_{m,\mu}$	Mean and SD of the GD for HFT MM thinking time	
T_m^i	Thinking time of HFT MM agent i where $T_m^i \sim \mathcal{N}(T_{m,\mu}, T_{m,\sigma})$	
V_i^c, V_m^i	Volumes of orders submitted by HFT SC and HFT agents, respectively.	

For/Dedicated to/To my...

PREFACE As such, this project turned out to be just as much about software engineering as The work presented in this thesis was the final project in

Part I

Searching model behavior

The model has several parameters which must be selected carefully before the simulation can be used to infer knowledge about market behavior.

The parameter tuning turned out to consume a significant amount of time, and simple using a genetic algorithm to optimize over the entire space of parameters was not enough. Instead, the process was a slow and iterative one of running the genetic algorithm to create a data set, analyze the data set to find out what was discovered in the search, and then run the genetic algorithm again with different parameters. Thus several data sets were created, each with the purpose of examining some aspect of the simulation, or of the parameter tuning method itself.

This chapter will cover the instruments used in the optimization of the model parameters, and also mention the machine learning tools used in the analysis of the data sets.

The parameter tuning has two overall goals, which are covered in the following section.

0.1 Selecting model parameters with inverse simulation

XXX The fact that the model is capable of reproducing market with dynamics which is unlikely tobe observed in real-world markets does not mean that the model is inherently bad.

Just as

Rather, since the model has a large number of parameters, the appropriate range of which cannot be determined a priori, markets

1. Define model behavior
2. Run the simulation with a wide range of different parameters
3. Select models which behaves in a way which is deemed realistic

XXX

Hence, inverse simulation is a way of transforming the problem of finding appropriate model parameters into a supervised learning problem. The next section will explain how the behavior of the model was summarized into a few simple performance measures that were used in the inverse simulation.

(A) Parameters causing unrealistic dynamics

(B) Unrealistic parameters causing realistic dynamics

FIGURE 1: **Motivatoin for tuning:** the two

0.2 Motivation and overall procedure

First of all, the model must be calibrated such that it mimics the behavior of real markets. Since virtually every aspect of the simulation behavior depends on the values on the various parameters, these must be chosen carefully in order for the simulation to produce realistic behavior. An example of a simulation untuned parameters causing unrealistic behavior is given in figure 1a. Selecting realistic parameters is by far a simple task. First of all, it requires a way of quantifying the quality of each simulation. The choice of such a quantification is discussed in section 0.3.2. Secondly, there might be several different parameter configurations which produce seemingly realistic behavior, but do not correspond to a realistic market setting. An example for this is given in figure 1b, and section 0.3.5 briefly discusses this point.

The second goal of the parameter tuning is to find parameters which promotes certain desirable behaviors. For instance, we might be interested in determining which parameters causes the traded price to stabilize faster after a shock to the fundamental price. Metrics for doing this is discussed in section 0.3.2

The selection of parameters is a fairly complicated process because of the large parameter space, and because it takes a significant time to evaluate the fitness of a given set of parameters[1]. amount of time to execute a simulation. Because of this, the following three-step parameter selection procedure was used.

1. Fix some of the model parameters in order to reduce the search space for the optimization algorithm. This requires us to consider which parameters can be

[1]The calculation time depends largely on the parameters, such as the number of agents and how active these are. Typically one to several minutes are required to evaluate a single set of parameters.

fixed without losing opportunity to gain insight into market behavior. Essentially this step is a question of prioritizing the optimization of some parameters over the optimization of others.

2. Use an optimization algorithm to find sets of parameters which yield realistic model behavior. A genetic algorithm was chosen for this purpose, and the details are explained in section 0.3.

3. From the set of parameter combinations found by the optimization algorithm, remove the parameter combinations which obviously do not correspond to a realistic setting.

0.2.1 Selecting fixed parameters

The main parameters of interest are the ones that control the latency and speed of the agents. The agent strategy parameters are less important, since

Number of rounds Due to the computational cost of running the simulation for a large number of rounds, the the number of rounds is fixed at 10^5 for all experiments.

HFT order volumes When the HFT agents are initialized in the beginning of the simulation, each agent is assigned with a constant, sampled from a normal distribution, designating the fixed volume of the orders that the agent will submit throughout the simulation. Hence, market maker MMi will submit orders with a volume of V_m^i, where $V_m^i \sim \mathcal{N}(V_{m,\mu}, V_{m,\sigma})$, and chartist SC$j$ will submit orders with volume V_j^c, where $V_j^c \sim \mathcal{N}(V_{c,\mu}, V_{c,\sigma})$. XXXMARKET MAKERS SUBMIT LARGER ORDERS THAN CHARTISTSXXX

Each HFT agent submit orders with a volume that is constant throughout the simulation.

Average arrival rate of slow trader orders

Slow trader order volumes

Agent start portfolio All HFT traders start with a fixed amount of cash and a fixed number of shares in their portfolio.

Order book initialization In all simulations, the order book is initialized with $10 * *5$ orders with prices sampled from the distribution $\mathcal{N}(F0, 10 * *2)$ and volumes sampled from $\mathcal{N}(20, 5)$.

The remaining model parameters will either be fixed for each experiment, or varied by the genetic algorithm.

0.3 Inverse simulation with a genetic algorithm

Inverse simulation refers to the technique of specifying metrics measuring model behavior, and then using an optimization algorithm to search for parameters resulting in desirable (or undesirable) behavior.

In this work, a genetic algorithm was used to search the parameter space. The algorithm proceeds as explained below.

1. Generate a population of healthy individuals, e.g., individuals with valid parameters.
2. Evaluate fitness for every individual in the population.
3. Repeat n_{gen} times
   (a) Generate offspring by crossing existing individuals.
   (b) Apply mutation to with a certain probability to each individual (parents as well as children)
   (c) Evaluate fitness of children and mutated parents.

Mutation and crossover are the operators responsible for generating variation in the population, while the selection is responsible for propagating promising individuals to future generation where they might be improved. Several possible methods of performing each of these three steps exist in the literature (see[**?**], [**?**]), and section 0.3.3 briefly covers the method and parameters of the genetic algorithm.

0.3.1 Representing parameters as genes

Since the choice of mutation and crossover operators depends on the nature of the genes, the first step towards utilizing the genetic algorithm to search the model parameters is to decide on how to encode the parameters as individuals. A set of parameters is represented by an individual, i, consisting gene for each parameter, represented by a floating point $g_{i,j}$, where j denotes the index of the parameter. When the population is initialized, each $g_{i,j}$ is drawn from a uniform distributed in the range $g_j \in [0; 1]$:

$$g_{i,j} \sim \mathcal{U}(0, 1) \tag{1}$$

Some of the model parameters are integers, such as N_m and N_c, and these are rounded after being scaled, and before they are passed to the simulation.

0.3.2 Model fitness

XXX NOT FINISHED YET. In order to use inverse simulation, it is necessary to decide on how to measure the quality of an instance of the simulation. In this work, the overall goal is to examine which parameter values cause the market to be stable, and which cause it to be unstable. In order to do this, four fitness measures were defined as below. Each of the fitness measures are designed to reflect different properties of the simulation, but some correlation between the measures were found. This is discussed in section **??**.

0.3.2.1 Market response time

After the negative shock to the fundamental price, the slow fundamentalists will start submitting orders at lower prices, thus creating a force that drives that traded price towards the new price. The number of rounds that elapse before the asset is traded at the new fundamental can be thought of as the response time of the market and this property of the market is interesting for a few reasons. If a market has a slow response time, it means that the asset is traded for more than it is worth for a longer of period of time.

The response time is denoted f_{t} and is calculated as follows:

$$f_{\mathrm{t}} = \arg\min_{T} \left\{ p^t_{\min}(T) \right\},\ \forall n > t_\eta \tag{2}$$

where $p^t_{\min}(T)$ is the price of the order traded at the lowest price in round n:

$$p^t_{\min}(n) = \min \left\{ p^t_1(n),\ p^t_2(n), \ldots,\ p^t_K(n) \right\} \tag{3}$$

If the trade price never reaches the new fundamental, f_{t} is undefined.

0.3.2.2 Market overshoot

In some markets the traded price never becomes smaller than η, while in other markets the share price overshoots the new fundamental by being traded at prices lower than η. In the case that $\eta < 0$, the overshoot reflects an under-valuation of the stock.

The market overshoot is calculated as the number of ticks between the traded price and the fundamental price, in any of the rounds following the round in which the stock is traded at price $p^t_i(n) = F\eta$ for the first time, denoted tF:

$$f_{\mathrm{o}} = \mid \max p^t_i(n) \mid,\ \forall i,\ n > tF \tag{4}$$

f_o does not measure the time duration of the under-evaluation.

0.3.2.3 Price flickering

This fitness measure reflects how much the traded price fluctuates by calculating the standard deviation of the prices of all trades executed after tF:

$$f_\sigma = \sum_{n=tF}^{\mathtt{N_T}} \sum_{i=1}^{N_n} \frac{\left(\bar{p}_n - p_i^t(n)\right)^2}{N_n} \tag{5}$$

where N_j is the number of transactions in round j, and $\bar{p}_n$ is the average traded price in round n.

Time to stabilize

The margin of two ticks sets a fairly strict requirement for when a simulation is considered stable, as a flicker of just a few ticks

$$\arg\min \forall \tag{6}$$

Figure XXX illustrated how the four fitness measures are calculated, while figure XXX gives four examples and lists the respective fitness values calculated for each simulation.

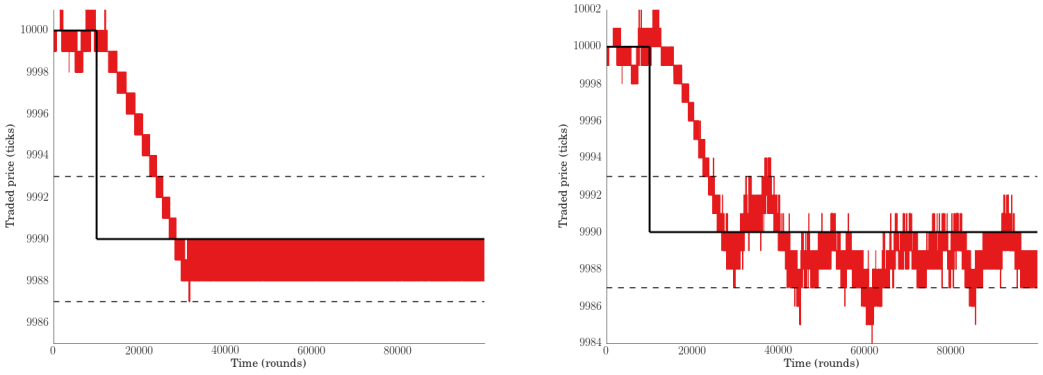
Typical cases

To give the reader a more intuitive idea of how the fitness measures summarize the model behavior, figure 2 gives six examples of some fairly common patterns.

Relationship between $f_\mathbf{t}$ and $f_\mathbf{s}$

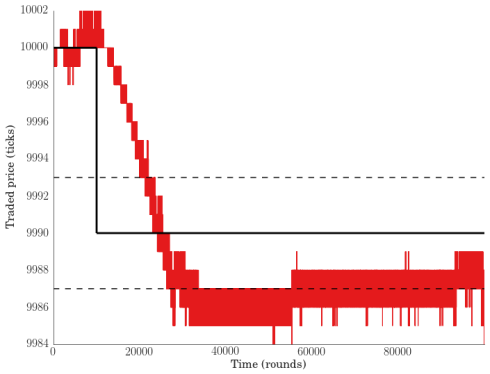
$f_\mathbf{s} < f_\mathbf{t}$ This happens when the traded price never leaves the stability margin after reaching the new fundamental price. Note however that this case does not necessarily mean that the prices do not flicker.

$f_\mathbf{s} > f_\mathbf{t}$ This happens when the traded price leaves the stability margin once or more after reaching the new fundamental. The traded price can be close to the fundamental, but flickers in and out of the stability margin as on Figure 3a shows an example where the trade price fairly stable and with no overshoot, leading to good (low) $f_\sigma$and $f_o$fitness values to be assigned to the parameters. However, even

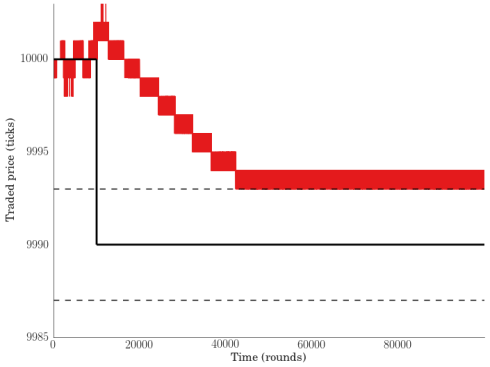


(A) Stable within margin

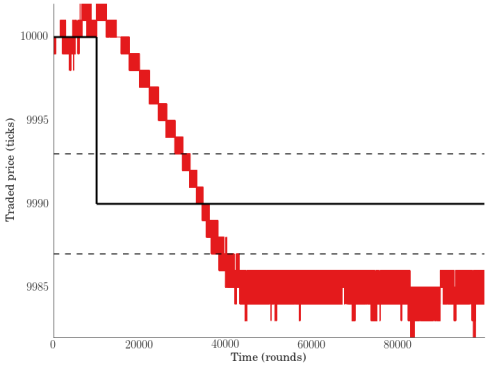
(B) Within margin, but with flickering



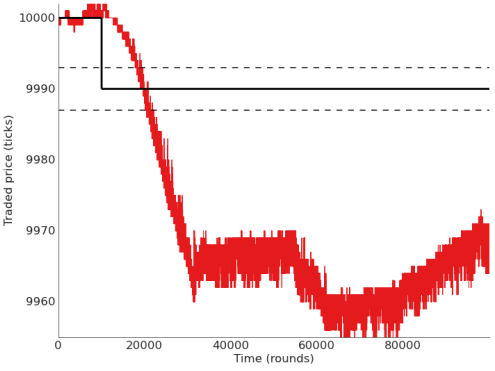
(C) Flickering on the edge of margin



(D) Never reach fundamental

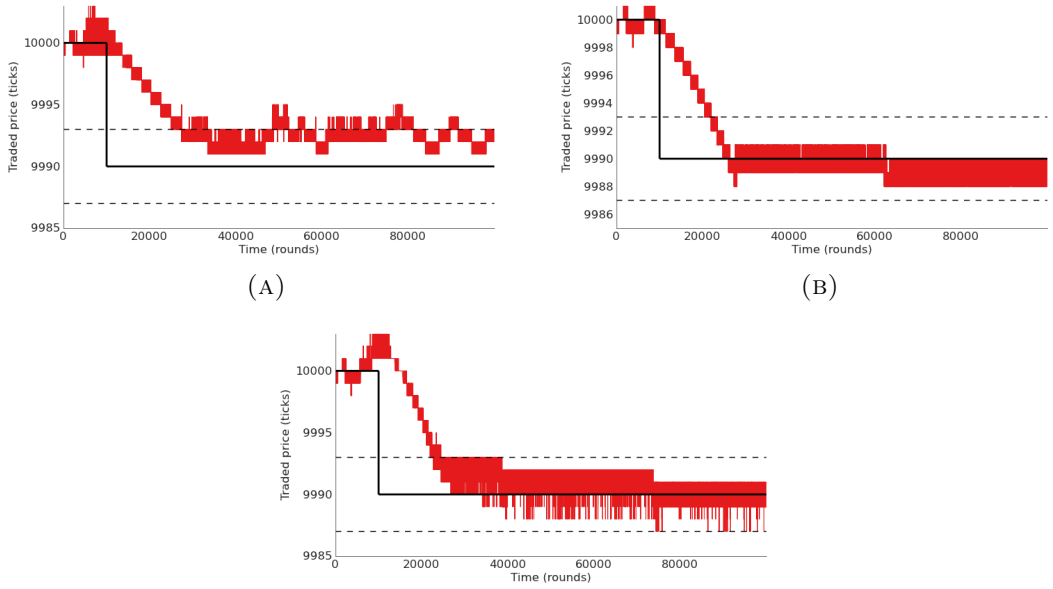


(E) Stabilize under fundamental



(F) Market crash

FIGURE 2: asdasd



(A) (B)

(C)

Parameter	Assignment
Number of individuals	200
Cross-over points	2
Tournament size	3
Mutation probability	0.1
Mutation distribution	$\mathcal{N}(\mu = 0, \sigma = 0.1)$

TABLE 5: Overview of parameters used in the genetic algorithm

though the traded prices are mostly within the stability margin, occasional flickers out of the margin causes the simulation to score a bad (high) f_{s}fitness. Note also that $f_{\mathrm{t}}$is undefined in this case.

$f_{\mathbf{s}} = f_{\mathbf{t}}$ This happens if a trade is executed at price $m_{\mathrm{stable}} - p_{\mathrm{fas}} < p^m < m_{\mathrm{stable}} + p_{\mathrm{fas}}$, and another trade is executed at price $p^m = p_{\mathrm{fas}}$ in the same round.

0.3.3 Configuring with genetic algorithm

Although this is a basic version of genetic algorithm, using it correctly is not necessarily easy, as was encountered. First of all, the parameters for the genetic algorithm itself must be established. The larger and more complex the search space, the more resources the search will require, since the evaluating the fitness function (i.e., the running the simulation) will have to be done a larger number of times.

Table 5 presents an overview of the parameters used in the genetic algorithm.

	f_o	f_s	f_σ	f_t
Overall stable market	min	min	min	min
Crash	max	-	-	-

TABLE 6: Optimization criteria for different types of market behavior

0.3.4 Optimizing towards desired behavior

XXX WRITE MORE. The four fitness measures defined in section 0.3.2 make it possible to specify the type of market behavior that is likely to be selected by the genetic algorithm.

First of all, we are interested in establishing which parameters cause the market to return to a stable state after the fundamental price has incurred a shock

0.3.5 Filtering parameters

As mentioned earlier, it is not enough simply to define a fitness function which assigns high values to parameters causing realistic behavior. In addition, it is important to discard parameters which obviously do not correspond to a realistic setting. Imagine that the simulation scores high fitness values when executed without any market makers. Since it is known that real markets do in fact contain market makers, nothing can be inferred from such a result. Indeed this might be a consequence of poorly designed fitness measures, but since it is easier to use domain specific knowledge to filter out the unrealistic parameters

0.4 Applying the genetic algorithm

Applying the genetic algorithm to produce markets with desirable behavior turned out to be more difficult that one could have hoped for. First of all, the high computational costs was a hurdle.

Secondly, many of the data sets (see section 0.4.3) produced did not contain any useful information.

The model parameters do influence the fitness values as they control model behavior, but they are not directly weighted into the fitness-values. This means that even after the parameters of the genetic algorithm was properly tuned in such a way that higher-fitness individuals were produced, these individuals often turned out to be of no interest. Such

FIGURE 3: Example of a simulation which is assigned fairly good fitness values, but which was executed with clearly unrealistic parameters: $N_m = N_c = 0$. The simulation reaches the new fundamental price fairly quickly without any undershoot, and stays within the stability margin. The only point where it scores badly is the standard deviation which is slightly high due to the fluctuating trade price.

individuals were discarded according to the filtering criteria described in section 0.3.5 would have to be discarded. An example of such a case is discussed in section **??**

0.4.1 Time complexity

The high time complexity stems from several factors

- Running a simulation for a given set of parameters required up to several minutes of computation on a single CPU core.
- Large number of model parameters increase the size of the search space. The more parameters Unfortunately there is no magic to the way the genetic algorithm works, and optimizing in a larger search means a larger time complexity.
- Large range of parameters. Some of the parameters are integers while some are real numbers. While most of the parameters have a lower bound, none of the parameters have upper bounds.

- Unstable fitness parameters. Since the same set of parameters can produce varying model behavior, the fitness-values may also vary. Therefore it is necessary to evaluate the simulation several times for each set of parameters.
- The model parameters also influence the time complexity. For instance, evaluating a simulation with many agents takes more time than evaluating a simulation with fewer agents.

Genetic algorithms are naturally suited for parallel computation. Two servers with a total of 40 cores and enough memory to evaluate as many simulations were utilized. With this equipment, evaluating a single strategy (see section 0.4.2)for generating data sets could take

reaching a point where the genetic algorithm began to produce useful output turned out to be somewhat of an iterative process.

First of all, the computational cost of running the genetic algorithm was high, which meant that the algorithm was not always able to find better individuals.

Det tager lang tid -¿ faerre params skod resultater -¿ nye experimenter

Second of all, even when the genetic algorithm did manage to improve the fitness of the population, this did not always result in useful data.

0.4.2 Experiments: Dividing the search into parts

As mentioned earlier, the number of parameters and the range of each parameter influences the complexity of the search. Because of this, it is desirable to keep the number of parameters that are included in each individual as small as possible. However, fixing parameters means that some interesting properties about the model might not be discovered. Furthermore, varying all parameters at the same time makes the analysis and interpretation of the results more difficult. In an attempt to overcome this dilemma, several "experiments" were carried out [2]. Instead of trying to optimize all the model parameters at once, the search was split into several parts, each of which we call an experiment. Each of these experiments produce a data set, each of which were analyzed using the methods described in section 0.5. Some of the data sets produced interesting results, while others did not. Chapter **??** focuses on the analysis and presents the findings. A brief overview of the experiments is presented in **??**, but since the motivation for creating each data set is best understood in the context of the analysis of each data

[2]The reason for the quotes is that the term experiment might be stretching the common understanding of what an experiment is a little.

	$\lambda_{c,\mu}$	$\lambda_{c,\sigma}$	N_c	$T_{c,\mu}$	$T_{c,\sigma}$	$H_{c,\mu}$	$H_{c,\sigma}$	$W_{c,\mu}$	$W_{c,\sigma}$	$\lambda_{m,\mu}$	$\lambda_{m,\sigma}$	N_m	$T_{m,\mu}$	$T_{m,\sigma}$
0	84	11	14	98	9	1071	445	38	17	3	2	48	8	1
1	23	21	74	49	24	529	554	45	13	9	0	8	5	3
2	51	13	53	47	13	3586	536	10	11	9	4	14	4	2
3	18	21	213	70	39	793	1179	33	15	7	2	43	6	3
4	94	41	144	10	25	2668	893	12	15	6	1	49	7	4
5	19	4	130	15	38	1085	1165	39	4	2	3	11	4	4
6	65	15	91	81	46	3867	1991	48	1	7	2	21	4	4
7	36	38	143	77	19	2805	1870	10	9	7	0	3	2	4
8	43	8	10	19	19	3384	1706	33	4	8	4	5	5	0
9	11	33	127	94	49	3597	723	12	2	7	1	33	5	4

TABLE 7: An example data matrix containing the parameters of ten individuals who lived sometime during the execution of the genetic algorithm. In this case, each individual contained parameters for the number of HFT agents, as well as the latency and thinking time parameters. Hence, the data matrix has a column for each parameter.

set, the details are deferred until chapter **??**. The next section will explain exactly what a data set is.

0.4.3 Gene pool as data set

The previous sections contain the details of each of the steps undertaken in order to produce data sets. To summarize, the list below enumerates the steps.

1. Initialize a population in the genetic algorithm with healthy individuals.
2. Evaluate the fitness for every individual several times and obtain fitness-values by calculating averages.
3. Stack all individuals that ever lived into a $N \times \mathcal{L}_i$ parameter data matrix $\boldsymbol{P}$, where N is the number of individuals, and $\mathcal{L}_i$ is the length of each individual. Likewise, stack the fitness values into a $N \times \mathcal{L}_f$ fitness-data matrix $\boldsymbol{F}$, where $\mathcal{L}_f$is the number of fitness values calculated.
4. Filter the data by removing rows in $\boldsymbol{P}$with parameters which can be deemed not to correspond to real markets, and by removing rows in $\boldsymbol{F}$with fitness values that are not realistic. Please refer to section 0.3.5 for details. Naturally, when a row is removed in $\boldsymbol{P}$, it is also removed in $\boldsymbol{F}$, and vice versa.
5. Likewise, data points which were generated by a simulation crashing before it could complete were removed.

Tables 7 and 8 contain the first rows of $\boldsymbol{P}$and $\boldsymbol{F}$for one of the data sets.

	f_o	f_s	f_σ	f_t
0	3	25359	0.382092	29838
1	7	99999	1.289659	23373
2	6	99999	1.253363	18748
3	7	99997	1.695150	22819
4	6	94343	1.329276	22703
5	16	99999	2.439084	31860
6	6	93378	1.287235	25645
7	10	99997	1.858166	19417
8	3	24039	0.935465	27381
9	19	99995	4.092439	24845

TABLE 8: This table contains the fitness values for each individual in table 7. Note that, in order to increase the reliability of the fitness measure of an individual, the recorded fitness-values are the average of the fitness-values obtained by evaluating each individual ten times

0.5 Data analysis

Using inverse simulation merely creates a lot of data. This data has to be analyzed before any

Data normalization

0.5.1 Data visualization

It is useful to be able to plot the data points

0.5.1.1 Color toned scatter plots

Scatter plots are useful for initial data analysis, as one can quickly detect problems such as outliers, and maybe even detect clusters of data points.

Scatter plots are probably among the most rudimentary of techniques for data analysis, yet they can be incredibly informative, especially when the data that is visualized is low-dimensional. The two plots in figure make two things clear. First of all, extreme values occur in f_σ. Even log scaling does not seem to fix this problem entirely. Second of all, XXXissue109XXX

0.5.2 Preprocessing

0.5.2.1 Handling outliers

The term outliers is often used as a label for data which is considered "invalid" in the sense that is not a product of the true data generation process, but due to various sources of noise. In this report, data that was caused by failing simulations corresponds to the usual understanding of outliers. However, as was already explained in section **??**, data from such simulations are never included in $\boldsymbol{P}$ and $\boldsymbol{F}$ to begin with. Instead, outliers in this report refer to entries in $\boldsymbol{P}$ and $\boldsymbol{F}$ deviate significantly from the majority of the data points by having extreme values.

Such data points caused problems when applying data analysis techniques which rely on a fairly normal distribution of the data points, such as Principal Component Analysis (PCA). PCA looks for a rotation of the data space, such that the axes of the new basis are aligned with the directions of the largest variance in the original space. Since a few points with extreme values come to account for a large portion of the data set variance, the new basis computed by PCA will be aligned along these few data points. When PCA is used to extract lower dimensional features from the data set, outliers will degrade the quality of these features. In the case that PCA is used for data visualization, the scatter plots of the first few principal components will not be very informative, as they merely show the projection of the data onto the axes aligned with the outliers.

In all experiments, outliers were present in both the parameters space and in the fitness space. Outliers in the parameter space occur because of abnormally large mutations, and any dimension of the parameter space is susceptible to such an event. In the fitness space, only a few of the features suffered from the occurrence of outliers. The feature f_{s}cannot contain outliers, because the shock to the fundamental occurs at round $t = 10^4$, and because the simulation is terminated at round $t = 10^5$, hence $f_{\mathrm{s}} \in [10^4, 10^4 + 1, \ldots 10^5]$. The same is true for f_{t}. $f_\sigma$and f_{o}, on the other hand, are susceptible to outliers, because the two features have no upper bound.

1. Apply a monotonically increasing transformation $f(x)$ to some or all of the features for every data point in the data set. A common choice of f is $f(x) = \log x$, as it efficiently reduces the impact of data points with extremely high values. Figure **??** illustrates the effect of applying the log-transform. The log-scaling does a fairly good job of reducing the importance of the outliers in the $f_\sigma$feature, while the change is less dramatic in f_{t}. While the left scatter plot does not really reveal any structure of the data, the transformation makes it possible to spot to rough clusters when inspecting the right plot.

Another simple method is to manually remove

2. Manually select one or more criteria for when a data point is to be considered an outlier. It is worth noticing that $f_{\text{o}}$and f_{σ}

3. Use a one-class support vector machine to calculate a probability for each data point that said point is an inlier or an outlier. In this case, inliers

In the parameters data set, outliers can occur due to abnormally large mutations. Some parameters cause the simulation to act in strange ways, and even crash in some cases. For instance, the the order book becomes empty, the simulation throws an exception and terminates. Similarly, if the best bid/askprices drops to zero, the simulation exits. As such

The most common odd phenomenon was the market

0.5.3 Clustering algorithms

0.5.3.1 GMM

covariance type:full

Appendix A

Additional tables

A.1 Dataset 1

Write your Appendix content here.

Appendix B

Third party software

Deap scoop sklearn matplotlib numpy, scipy and pandas geometric brownian walk

Appendix C

Additional figures

Bibliography

[1] C. J. Hawthorn, K. P. Weber, and R. E. Scholten. Littrow configuration tunable external cavity diode laser with fixed direction output beam. *Review of Scientific Instruments*, 72(12):4477–4479, December 2001. URL http://link.aip.org/link/?RSI/72/4477/1.

Bibliography

[1] Chi Wang, Kiyoshi Izumi, Takanobu Mizuta and Shinobu Yoshimura: "Investigating the Impact of Trading Frequencies of Market Makers: a Multi-agent Simulation Approach" *SICE Journal of Control Measurement, and System Integration, Vol. 4, No. 1, pp. 001-005, January 2011.*

[2] Michael J. McGowan: "The Rise of Computerized High Frequency Trading: Use and Controversies", *2010 Duke L. & Tech. Rev., 2010.*

[3] Thomas McInish and James Upson: "Strategic Liquidity Supply in a Market with Fast and Slow Traders", *Available at SSRN 1924991 (2012).*

[4] Niel Johnson, Guannan Zhao, Eric Hunsader, Jing Meng, Amith Ravindar, Spencer Carran and Brian Tivnan: "Financial Black Swans Driven by Ultrafast Machine Ecology", *Available at SSRN (2012).*

[5] Kiyoshii Izumi, Kiyoshi, Fujio Toriumi, and Hiroki Matsui: "Evaluation of automated-trading strategies using an artificial market", *Neurocomputing 72.16 (2009): 3469-3476.*

[6] Chiarella, Carl, Giulia Iori, and Josep Perelló: "The impact of heterogeneous trading rules on the limit order book and order flows" *Journal of Economic Dynamics and Control 33.3 (2009): 525-537.*

[7] Peter Gomber, Björn Arndt, Marco Lutat, Tim Uhle: "High-frequency trading." *Available at SSRN 1858626 (2011).*

[8] J. Doyne Farmer and Spyros Skouras: "An ecological perspective on the future of computer trading" *Quantitative Finance 13.3 (2013): 325-346.*