Surveillance of algorithmic trading

Rev 1.2 January 16, 2013

A proposal for how to monitor trading algorithms

Lars-Ivar Sellberg
Executive Chairman, Scila AB
MSc Electrical Engineering/IT
BA Financial Economics

Table of contents

1 Monitoring trading algorithms using boundary conditions

1.1 Foreword

The author would like to thank Johan Wattenström who is an independent trader with long experience in algorithmic trading and Sergei Mayorov at Micex-RTS for their valuable comments to this paper.

1.2 Introduction

As various types of algorithmic trading makes up an increasing part of the trading volume on stock exchanges and other execution venues, it is generally acknowledged that these algorithms provide valuable liquidity that would otherwise not be available. At the same time concerns are being raised that algorithmic trading does have negative aspects besides its positive effects. The main concerns that are normally referred to can be summarized as:

- Increased volatility, leading to temporary illiquidity during periods of market stress. A widely known example is the “flash crash” of May 6, 2010 involving a negative feedback loop, where different types of algorithms are interacting with each other ultimately producing a situation of severe lack of liquidity.
- Market instability due to runaway algorithms disrupting the price discovery process and generating excessively large transaction flows. The event of August 1, 2012 in the US markets involving Knight Capital is such an example.
- New forms of market abuse, an example being the much debated practice of quote stuffing.
- Lack of fairness in terms of market access between market participants. The cost of infrastructure able to compete on equal terms with cutting edge HFT firms is not manageable for a large portion of the market participants.

  While this item normally is a hot topic in many discussions concerning the drawbacks of algorithmic trading, it tends to be less of a practical problem in mature electronic markets. Once enough firms have invested in the required technology the most obvious arbitrage opportunities disappears and spreads narrows eliminating excessive profit opportunities.

While the reactions from regulators and legislative bodies have been slow in coming, it has now virtually exploded in a plethora of suggested and implemented counter measures to these perceived negative effects of algorithmic trading. The problems with a majority of these countermeasures are that while they often provide at least a partial solution to the problem, the side effects are poisonous enough to kill the patient, i.e. the market. Some of the more discussed countermeasures include:

1) Minimum resting times. Imposing a minimum time order needs to reside in the order book, and be available for matching. While superficially appealing, it would create opportunities for new set of algorithms to take advantage of stale orders left behind when the market is shifting. This would also imply that market makers would

have to widen their spread in order to reduce the risk of having stale quotes in the market.

Indeed a variant of minimum resting times can have said to been tried in some specialist based markets in the US. Incoming orders were frozen for a certain time in the specialists book allowing the specialist to do arbitrage between the frozen order and a fully automated markets elsewhere. The end result was not positive for the owners of the frozen orders.

2) Market maker obligations. Users of algorithms are required to provide two-way prices for the entire trading day. The idea stems from the fact that a lot of HFT algorithms provide two-way prices, but unlike market makers, HFT users are not obliged to provide such prices for the entire day. Forcing them to do so without providing any sort of compensation is likely to drive a significant amount of HFT liquidity away from the market.

   While the practice of market making exists today in some markets for less liquid stocks, the market maker is paid by the listed company in order to increase the liquidity in their share. Even in small stocks with larger spreads and a fee this is today a business that traders rather do without and it's usually a service to the client as a part of a relationship aiming to gain corporate finance business from that company.

3) Charging for excessive amounts of generated messages i.e. a high order to trade ratio. Adding an extra trading fee if an trading algorithm is generating an excessive amount of messages do have a general long term dampening effect but is unlikely to hinder negative feedback loops causing damage in short term.

4) Notification of algorithms. Before deploying an algorithm it must be registered and fully explained to the regulator or owner of the market venue. If successfully implemented it would be a great advantage, but in reality the cost of not only documenting the inner workings of the deployed algorithms, but to actually transfer that knowledge to the regulator is prohibitive.

   Consider, that if the regulator is to have any real practical use of the information provided by the owner of the algorithms, they must be able define market scenarios and then study how the algorithms behave in each such envisioned scenario. Also the owner of the algorithm might have an issue with revealing the inner workings of the algorithm.

   In addition, algorithms are always a work in progress, as the market is always changing, the algorithms must be constantly improved and adapted. The time to market for an algorithm with a notification process would often render the algorithm obsolete before it is even accepted.

The purpose of this paper is to suggest a variant of counter measure 4 “Notification of algorithms” that would provide most of the benefits of this method, while drastically reducing the excessive cost associated with the original method.

The basic idea is for the owner of the algorithm to define and provide the operator of a market venue not with the inner working of the algorithm but a set of boundary conditions, that is expected to hold true at all times. These boundary conditions are then monitored instead of the inner workings of the algorithms. Some trivial but still relevant examples of boundary conditions would be:

- “This algorithm will not produce more than 5000 transactions/second at any time”
- “The maximum turnover per 10 seconds is 100.000 Euro”

Combining several, by themselves, relatively simple boundary conditions can yield a finely meshed net capable of detecting a wide range of deviant behavior. Indeed, this is a well-established method within the software testing community for detecting software bugs during simulation runs.

From an implementation point of view, designers of trading algorithms would be required to mark transactions with an id identifying the algorithm from which the transaction was generated. A surveillance application operated by the market place can then use these identifiers to calculate the current value of boundary conditions and compare them with the defined limits.

Once the infrastructure for monitoring the boundary conditions is in place, one could envision a process where the regulators define what type of boundary conditions that should be monitored, while it is up to operators to do the actual monitoring.

1.3 Advantages of the Monitor Algorithm Boundary Condition method

This method has several advantages

- It gives the regulator the freedom to impose what type of boundary conditions that should be mandatory to check, rather than leaving this to the judgment of the individual implementer of trading algorithms. Note that the regulator might not necessarily define the values for the boundary conditions, as opposed to the type of boundary condition, but instead leave that to the market place operator. This would enable the market place operator to fine-tune the values to suit his market model, trading participants and technical infrastructure.

- All owners of algorithms will report the values of their algorithm boundary conditions to the market places. The boundary conditions will include data, which will enable the market places to do proper capacity planning for their infrastructure. An example of such capacity related data is maximum number of transactions per second. This is an advantage in itself since improperly sized infrastructure unable to cope with the transaction flow is one source of market instability.

- The process of defining the boundary condition itself has a value in the sense that it forces the designer of the algorithm to consider what the boundaries of the algorithm are. Once again an analogy can be made with the software development community where conditions that should always hold true within a software module can be defined and monitored, using a mechanism called “invariants”.

One can envision that some designers of algorithms would protest and say, that it is not feasible to define boundary conditions of their algorithms. At that point regulators or operators of market operators would need to ask themselves if it is wise to deploy algorithms where even basic boundary conditions cannot be predicted and defined by the algorithm designer.

- Second opinion. Hopefully all implementations of trading algorithms have some sort of built in protection to prevent situations, where the algorithm is producing obviously erroneous trading decisions. It is quite likely that these internal safety measures take the form of boundary conditions checking the algorithm in a similar way to what is proposed by this paper. Why then re-implement them in an external application? If the implementer of the trading algorithm has made a mistake, i.e. introduced a software bug, the external application doing boundary checking will constitute a second layer safety net.

- The operator of the external application doing the boundary condition checking will have access to information not available to the owner of the algorithm. This information can then be useful for further boundary condition checking beyond the ones that have been given by the owner of the algorithm. An example of such information is the behavior of other algorithms.

  I.e. while the behavior of one given algorithm might be within the defined boundary condition and operating properly if looked upon in isolation, the situation might look different if one considers what other algorithms are doing at the same time. Negative feedback loops where algorithms are interacting with each other causing market instability have been identified and are one of the major culprits in several incidents. It would be possible for an application, monitoring the boundary condition of several algorithms at the same time, to detect if several of them are getting close to their limits at the same time, indicating some sort of a feedback loop

- Post-mortem event analysis. If some sorts of market instability event do occur, the monitoring and logging of boundary conditions will facilitate an analysis of whether some of the algorithms contributed to the event. Today these types of post-mortem analyses are often very time-consuming processes.

- Avoid revealing business secrets. While owners of algorithms are required to reveal the limits of their algorithms in terms of boundary conditions, these can scarcely be viewed as vital business secrets, especially if the alternative is to explain the entire algorithm to the regulator, which has been proposed elsewhere.

1.4 Suggested types of boundary conditions

The following is in no way an attempt to define an exhaustive list of relevant boundary conditions; its purpose is instead to illustrate how the method proposed in this paper can be used.

Examples of boundary conditions:

- Maximum rate of generated messages. Several time periods can be defined i.e. maximum messages/1 sec, maximum messages/10 sec etc.
- Maximum turnover per time period. Several time periods can be defined
- Maximum net position
- Maximum rate of spread changes. I.e. the rate at which the algorithm is allowed to change the best bid ask. Several time periods can be defined
- Maximum percentage of trading being done with a limited set of other algorithms as opposed to non-algorithmic orders. Several time periods can be defined. A rapidly increasing and high percentage might indicate the presence of a negative feedback loop.
- Minimum average resting time of orders. This is not to be confused with a general rule that all orders need to rest at least a certain time. This would be an average calculated for one or more time periods.

A regulator would most likely want to define a set of mandatory basic boundary conditions that have to be monitored. In addition to these basic conditions a market place operator might want to add additional conditions to be monitored.

There are two main reasons why a market place operator would like to add boundary conditions beyond what regulators mandate:

- Market integrity. Certain market actors, like the buy side, might be more inclined to access a market, given that they know that there are limits to how algorithms behave.
- Capacity planning. A market place operator always faces a tradeoff when adding a new trading participant who is using algorithms that, while providing liquidity, also consume a lot of resources in terms of technical infrastructure at the market place. Since trading participants often see resources at the market place as something they get for free, they tend not to optimize their algorithms to conserve it. On the contrary, it is common to see algorithms waste resources unnecessarily.

  Formalizing a contract in terms of boundary conditions restricting the consumption of technical infrastructure between the trading participant and market place operator makes it possible for the market place operator to take an informed decision whether the liquidity added by the algorithm is worth the cost of infrastructure.

1.5 Actions to be taken when a boundary condition is breached

Once regulators have defined boundary conditions and market place operators have calibrated boundary condition values to suit their market models, the remaining question is what to do once a boundary condition is breached.

Feasible alternatives include:

- Warn market place operator. The operators then take manual action, for example by contacting the trading participant or take whatever other action that is deemed necessary.
- Automatically reject any further orders generated by the algorithm and cancel all residing orders created by that algorithm.
- Impose a circuit breaker
- Publically flag the order book as being in a state of "fast market" warning other markets participants of unusual volatility.

While the optimal choice of action is dependent on the type of market model and other parameters it is unlikely that one given action should be the best choice for all situations. Rather a matrix of the following form could be used to define the action taken:

	Boundary condition, type A	Boundary condition, type B	Boundary condition, type C	Boundary condition, type D
Breach 10%	Action X	Action Z	etc.	
Breach 50%	Action Y			
Breach 100%	etc.			
Breach > 100%				

An interesting question is who is best suited to define what actions should be taken in a given situation, i.e. fill out the matrix above. It is certainly possible to argue that this should be a task for the regulator, since market place operators would be under pressure from trading participants to have as lenient rules as possible. On the other hand, the process of calibration, in order to meet the exact needs of the specific market place, is maybe more of a task for the operator of the market place.

Regardless of how much control the regulators chooses to exercise by them self in terms of defining types of boundary conditions, threshold values and actions taken when thresholds are breached, the fact that a process exists to monitor boundary conditions involving both market place operators and trading participants will in itself impose a certain level of discipline.

1.6 Conclusion

By defining and monitoring boundary conditions for trading algorithms it is possible to design an effective detection mechanism for misbehaving algorithms. While each monitored condition is relatively simple, the combined simultaneous checks of several conditions constitute a finely meshed net.

Types of boundary conditions to monitor could be mandated by both the regulator and the individual market place operator. The calibration process of setting thresholds values, and actions taken upon breach, might be most efficiently done at the market place operator level

The complexity of implementing the proposed solution is limited, thereby making it an attractive solution in an increasingly cost sensitive environment.