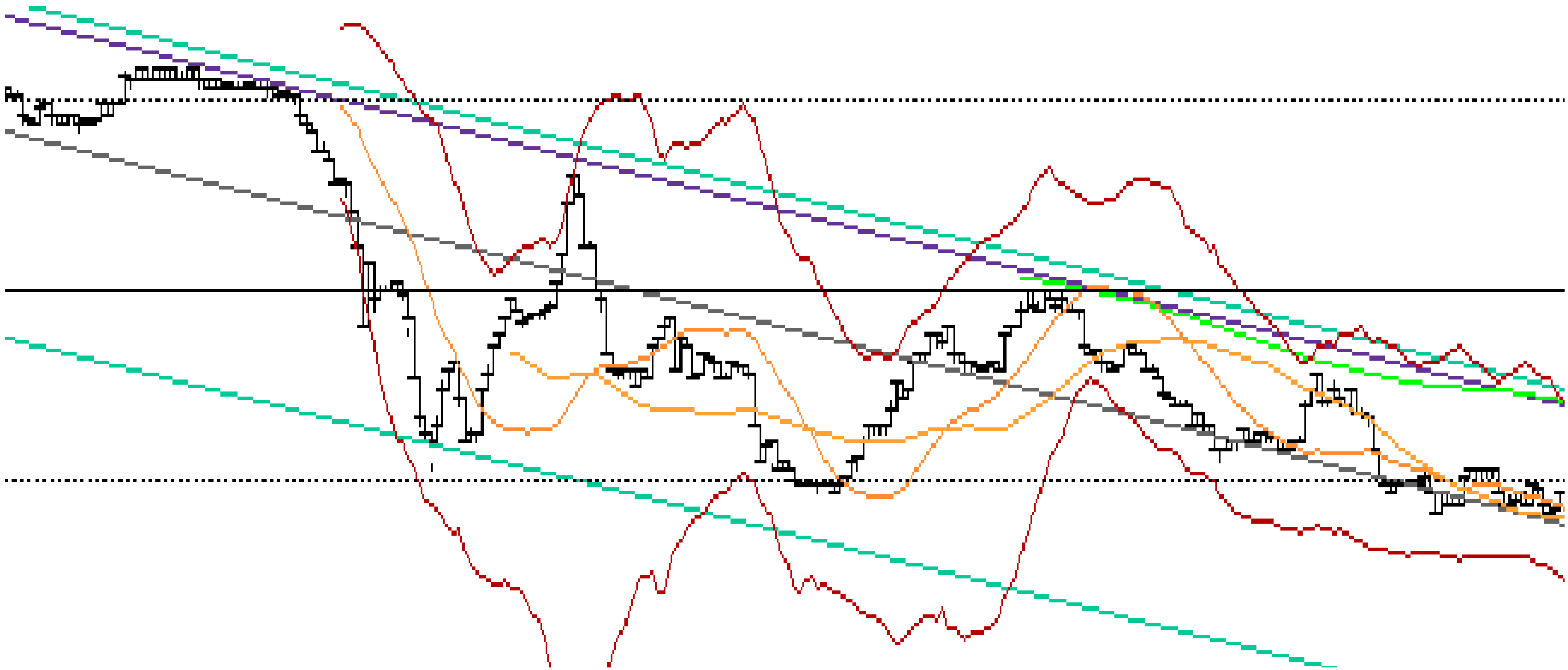
HTF
an alternative approach
by: Andrew Carpenter



What is HTF?

- High Frequency Trading :

  Most commonly known as trades taking place in time intervals ranging from hours to microseconds and the volumes of the stocks traded tend to be quite large ~ around 50,000 shares at a time.

- Additional HTF characteristics:

  Exploiting the inefficiencies of the market to make money off of the small fluctuations in price over a short time-interval

  Each individual stock sold usually only makes fractions of a dollar or even a single penny.

  HTF most often involves the use of an algorithmic trading strategy executed by computer programs written in c++

Key principles involved

- Most often the models for HTF algorithms make use of the inefficiencies of the market.

- “ The relative availability of trading opportunities can be measured as a degree of market inefficiency. ” [1]

- “ The more inefficient the market, the more predictable trading opportunities become available. Tests for market efficiency help discover the extent of predictable trading opportunities. ” [1]

- For inefficient markets, price fluctuations for a short period of time are have a degree of non-randomness and can be correlated to other factors within a certain degree of accuracy

Conclusion

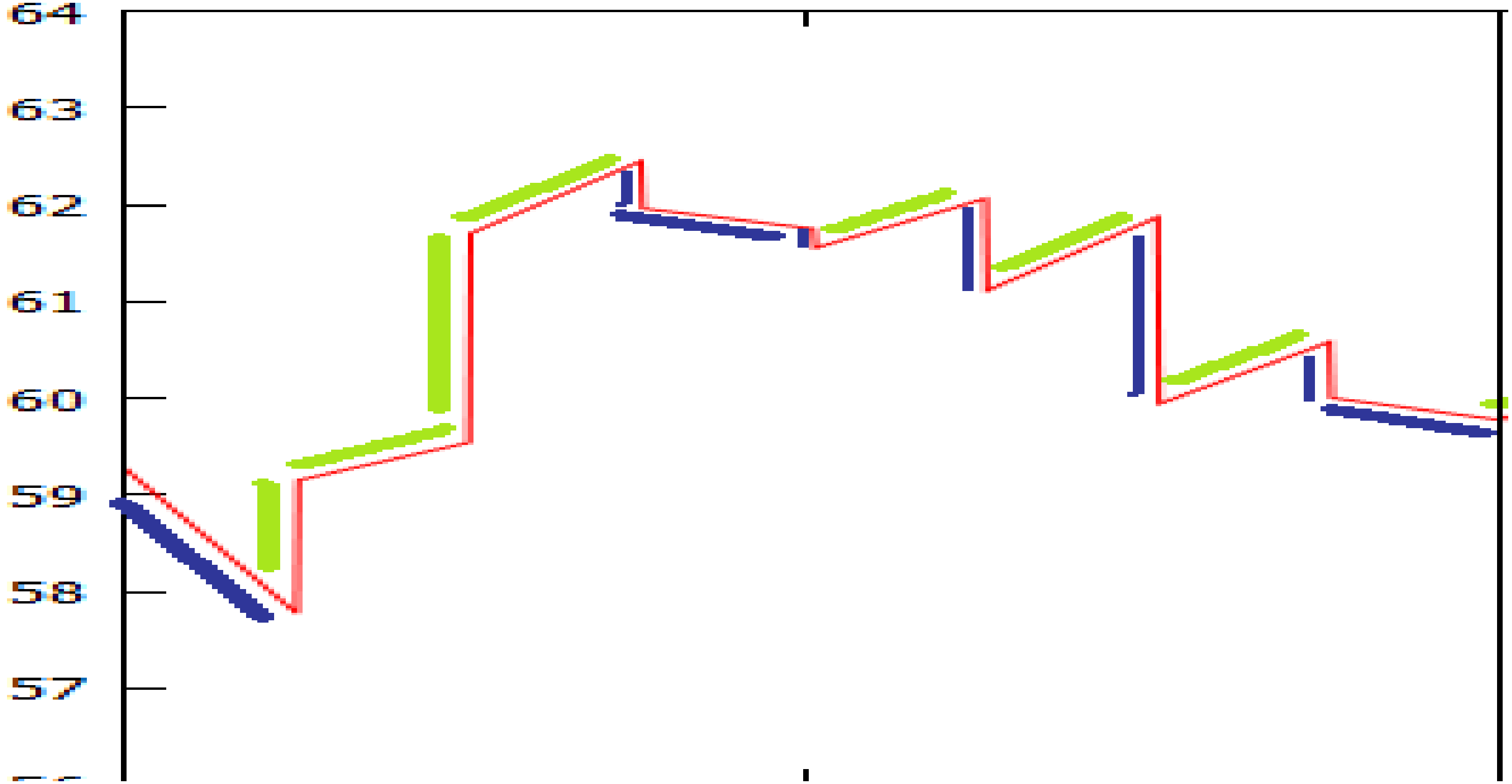
- If market is *inefficient*: price fluctuations are *predictable*.
- If market is *efficient*: price fluctuations are *unpredictable* ~ random.

An alternative approach

- If instead we look for efficient markets (markets for certain stocks) then we know that their prices should fluctuate randomly.
- These markets can be found by using certain *test* that check for randomness.
- The *opportunities* that exist in a randomly fluctuating market can be found by:

  Identifying the momentary local minimums in the price.

  Identifying the momentary local maximums in the price.



- n_1 = sum blue lines = 8
- n_2 = sum of green lines = 7
- u = sum of consecutive green lines and consecutive blue lines = 9

Test for randomness [1]

$$Z = \frac{|u - \bar{x}| - 0.5}{s} \qquad Z < 1.645$$

$$s = \sqrt{\frac{2n_1 n_2 (2n_1 n_2 - n_1 - n_2)}{(n_1 + n_2)^2 (n_1 + n_2 - 1)}} \qquad \bar{x} = \frac{2n_1 n_2}{n_1 + n_2} + 1$$

Denote the total number of runs, both positive and negative, observed in the sample as u

.

Denote n_1 as the number of positive 1-minute changes

Denote n_2 as the number of negative 1-minute changes

If $Z < 1.645$, then the 1-minute changes are random

Define local minimums:

- The condition that there is a *strong statistical chance* that the next *change* will be *positive*

  -The chance that you flip a coin to get 10 heads in a row is a bit small, therefore I'd be more willing to but my stakes on 7 heads and 3 tails.

- Determine a rule set in algorithm to define these favorable conditions to buy stocks

  -For example: Rule 1 – If n_2 increases 7 times consecutively, buy stocks at current price

A main for loop contains the Buy and Sell Rule set

For(i =1 ; i <362; i ++){ //i represents a minute

- Buy rule for-loop
- Sell rule for-loop

}

Use of Entropy – Buy Rule

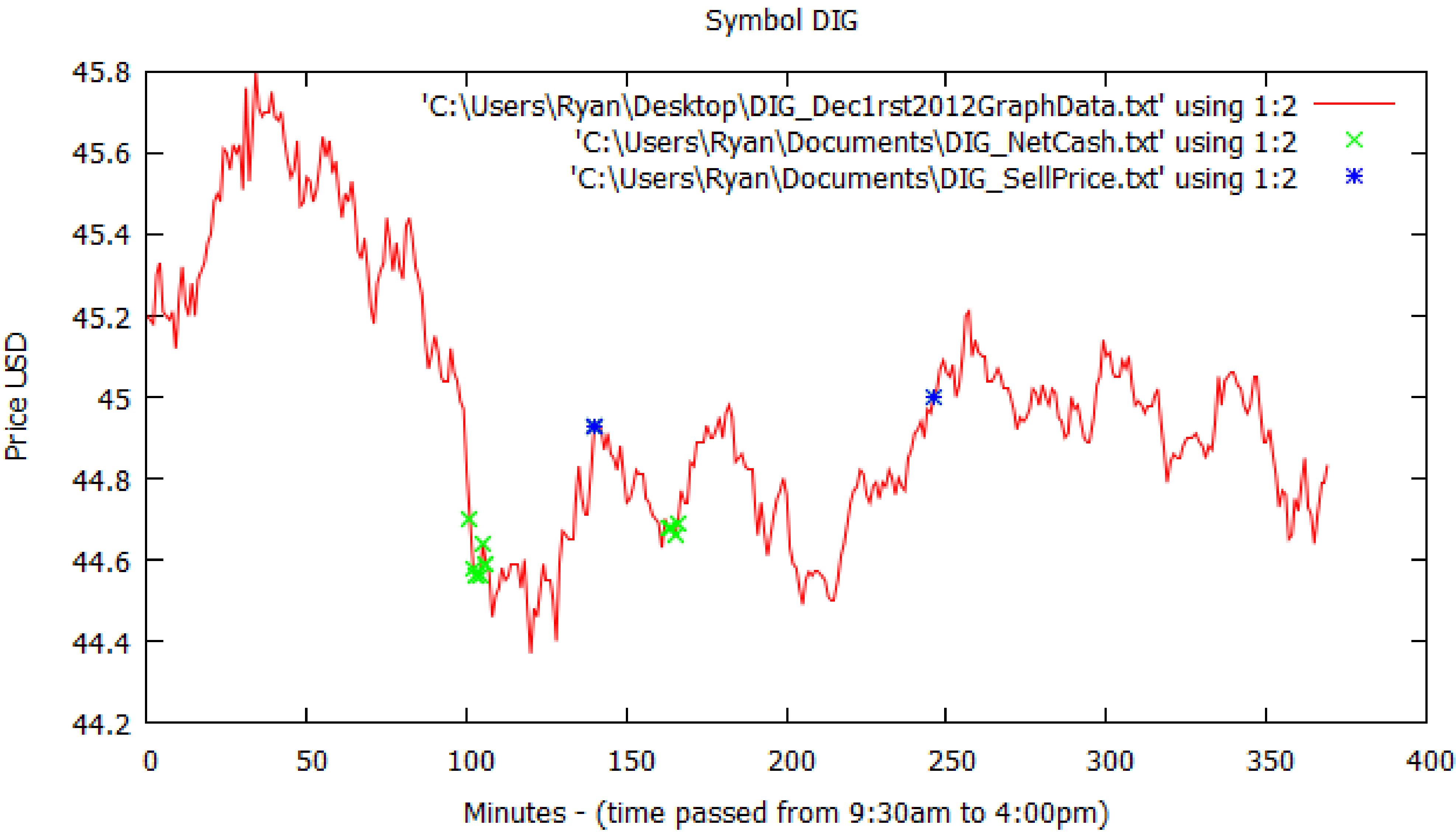
- U = Nn + Pn ;
- if(U > 0){probNn = (Nn / U);} //avoids singularities
- if(U > 0){probPn = (Pn / U);} // ^
- if(probNn > 0 && probPn > 0) {Entropy = -1.0*probNn*log10(probNn) + -1.0*probPn*log10(probPn);}

```
if( (1-pow( (Entropy/EnMax),2.0 ) ) > 0.8 && Nn > Pn && Nn > 9 && i < 362 &&
floor((netcash*0.5)/p[i]) >= 1.0 ){
        n++;
        nimax = n;
        boughtstocks5[n] = floor( (netcash*0.5)/p[i] ) ;
        netcash = (netcash - boughtstocks5[n]*p[i]) ;
        boughtprice5[n] = p[i];
        fout << i << "   " << boughtprice5[n] << endl;
        cout << i << "   " << "boughtprice5["<< n << "]=" <<
boughtprice5[n] << "   " << p[i] << endl;
        }
```

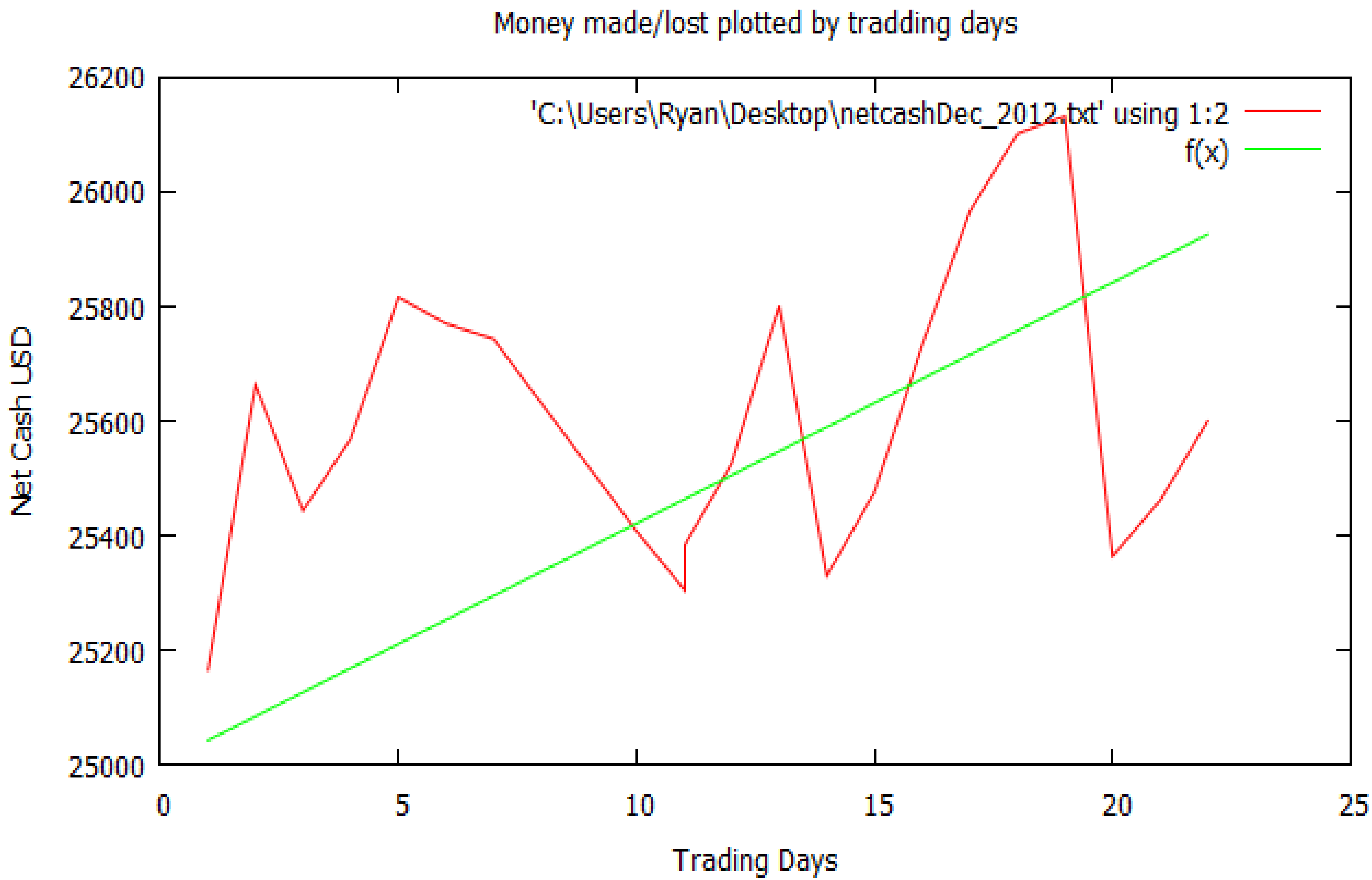
Sell Rule

- for(n=1; n <= nimax ; n++){
- if(p[i]*1.0 - boughtprice5[n]*1.0 > 0.01 && boughtstocks5[n] > 0 && boughtprice5[n] > 0.0){
- sellprice5[n] = p[i];
- netcash = netcash + boughtstocks5[n]*p[i]; boughtstocks5[n] = 0;
- cout << i << "   " << "sellprice5[" << n << "]= " << sellprice5[n] << endl;
- fout << i << "   " << sellprice5[n] << endl;
- }
- }

*Green points: Price bought at
*Blue points: Price sold at



Started off really well...



References

1. Aldridge, Irene. *High-frequency Trading: A Practical Guide to Algorithmic Strategies and Trading Systems*. Hoboken, NJ: Wiley, 2010. Print.