Short Summary About Current Design & Results

notations:

- **T**: time interval between time epochs
- **R**: the bouncing range of mean valuation in time epochs
- **N**: customer number
- **sig**: the sigma in Gaussian function. Smaller the sigma is, closer valuations are.
- **Ps**: the price of smart token in the market, updated after every transaction
- **Psc**: the price of smart token broadcasted by market at every beginning of time slot

1. Bancor Market

The whole simulating time is comprised of **1000** time slots. In every time slot, Bancor Market processes orders launched by **N** customers, and in the meanwhile alters the price of smart token.

Valuation Making:

At the beginning of every time slot, **N** customers will be announced about **Psc**, i.e., the current price of smart token. Based on this price, customers will generate valuations of smart token in Gaussian distribution with **mu** and **sigma.**
In normal cases, the **mu**, i.e., mean valuation in Gaussian is **Psc**, which indicates in normal cases, the ratio from valuations over the **Psc** to valuations below **Psc** is 1 : 1, i.e., around 50% customers launch buy orders while other 50% will create sell orders. However, in time epochs, which we simulate the market craze that at special cases most customers are eager to buy or eager to sell, the **mu** is randomly picked in range from **Psc/R** to **Psc*R.**

Code of random pick in time epochs:

```
if bool(random.getrandbits(1)):
    valuation_mu = random.uniform(currentMarketPrice/bouncingRange, currentMarketPrice)
else:
    valuation_mu = random.uniform(currentMarketPrice, currentMarketPrice*bouncingRange)
```

Transaction Generating:

After customers making their valuations of smart token, they will launch transaction orders with several stipulations:

1. If customers' valuation of smart token is larger than **Psc,** customers will launch transaction orders to buy the smart token; otherwise, sell orders will be generated.
2. If customers do not have reserve tokens in hand, they will not launch orders to buy smart token, though their valuations might be higher than **Psc**. Similarly, if customers do not have smart token to sell, they cannot generate sell orders.
3. Customers will make valuations and launch orders in every time slot; while in one time slot, single customer can only generate one order or does not generate, which indicates with **N** customers, at every time slot **<=N** transaction orders will be created. Therefore, 1000 time slots, totally **<=1000N** transaction orders will be made.

4. Customer uses all of their reserve tokens or smart tokens to buy or to sell if they have money in hand.
 * We also call 4 as all-in policy, which is implemented for simplicity.

The code for customers' transaction generation is presented as below:

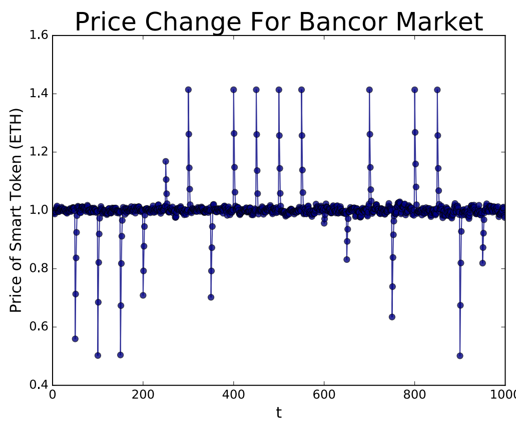
```
# self represents a customer (customer class). we name him/her as XXX
if self._valuation > self._market.getCurrentPrice() and self._reserveBalance > 0:
    # XXX issues a buy order
    self._market.buy(self, self._reserveBalance) # all-in policy
elif self._valuation < self._market.getCurrentPrice() and self._tokenBalance> 0:
    # XXX issue a sell order
    self._market.sell(self, self._tokenBalance)
else:
    # nothing to do
    pass
```

After customers generating their transaction orders by valuations and money they hold, they wait to see whether their transactions could be handled by market.

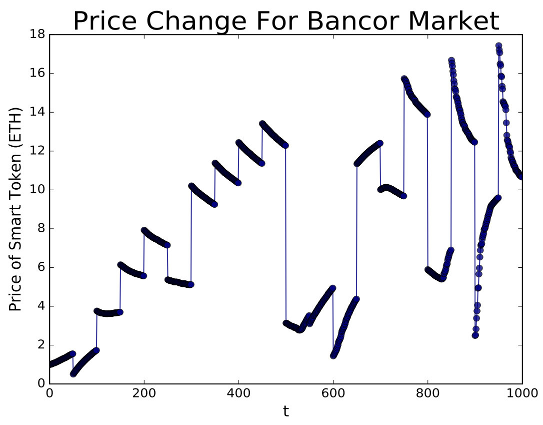
Transaction Processing:

Bancor Market processes customers' transaction orders **one by one**. In the meanwhile, it updates the price of smart token. Here, one by one means when dealing with one of customers' transaction orders, if the current price of smart token does not meet this customer's valuation, the market will announce the customer to cancel this order and skip this transaction order to try to deal with the next customer's order.

It might sound complicated, but the core idea is to simulate the limited market order in real life -- customers generate valuations of product first, and then accord to the product's price in market to decide whether finish the transaction. In Bancor Market, customers, based on the **Psc**, make their valuations while decide to finish the transaction by comparing **Ps** and valuations. (If the comparison tells customer he should not finish his order, this order will be canceled.)
Therefore, we record the price fluctuation of smart token as an indicator of Bancor Market's performance by figure 1.



(a) Without Reseve Aid



(b) With 200 Reserve Aid

Fig 1: The price change in Bancor Market.

The result in Figure 1(a) left us a puzzle last meeting. Unfortunately, I had not fully discussed this phenomenon clearly last time. I have sent you an email about the reason. It is all right if you have not read it yet. Please just read the problem1.pdf.

Further, since transaction orders might be canceled in market, we also track the cancel rate of transaction orders successfully launched by customers, which is plotted in Figure 2.

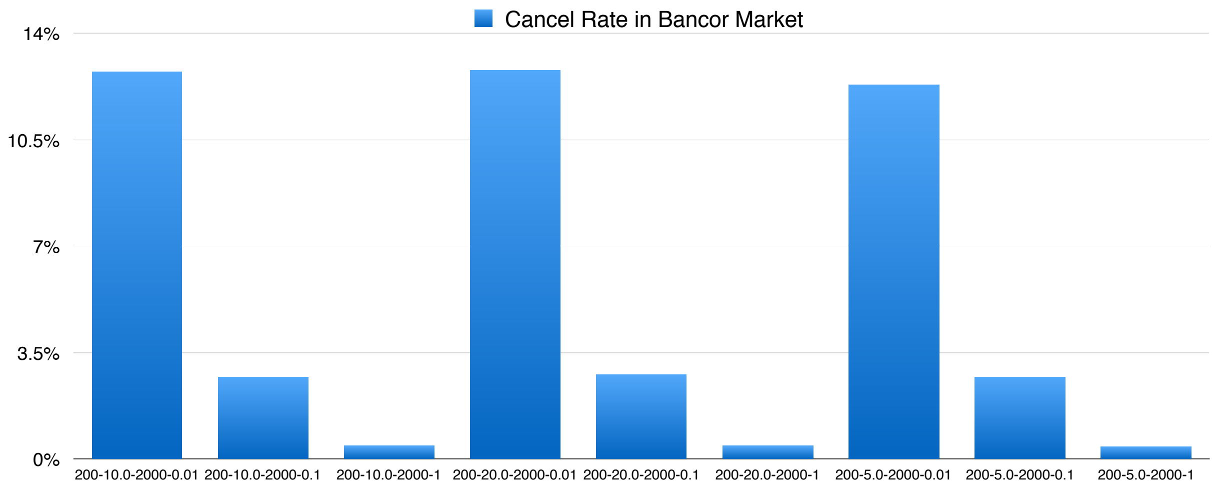


Figure 2. Figure about cancel rate in Bancor Market. Here the cancel rate is calculated by (# of all canceled orders / # of all launched orders). The x-axis: **T-R-N-sig**.

In Figure 2, what we can learn is that with smaller sigma, the transactions' cancel rate is much higher, which indicates more tightly customers make their valuations, more likely they need to cancel their transaction orders. The mathematical proof is delivered in problem2.pdf.

2. Classic Market

The whole simulating time is comprised of **1000** time slots. In each time slot, Classic Market processes the orders launched by **N** customers by managing the order list in the market.

Valuation Making:

The valuation making in Classic Market is similar with Bancor Market.
However, since the price of the product, i.e., **Ps** does not change, the mean valuation is constant in normal cases. That is that the **mu** always equals **Ps**.
Also, in time epoch, the **mu** is randomly picked in range (**Ps/R, Ps*R**).

Transaction Generating:

1~4 are similar with Bancor Market.
The only difference is that in Classic Market, the customer will not launch new order if his order has not been fulfilled.

For instance, in a certain time slot, a customer launches a sell order at valuation 5 ETH to sell all of his 200 smart tokens. However, in the next time slot, he finds that only 120 smart tokens have been sold. Therefore, he will not launch new order and continue to wait for his remaining 80 smart tokens being sold at valuation 5 ETH.
In the end of simulation, i.e., 1000 time slots have passed, if this transaction order is still unfinished in market, we say this order should be canceled.

Transaction Processing:

Classic Market manages an order list to process all transaction orders launched by customers.
In short, all transaction orders from customers will be separated into two sub-list, named as sell list and buy list. In each list, orders will be sorted by the valuation of these orders. Then the market processes these orders by methods shown in Figure 3.

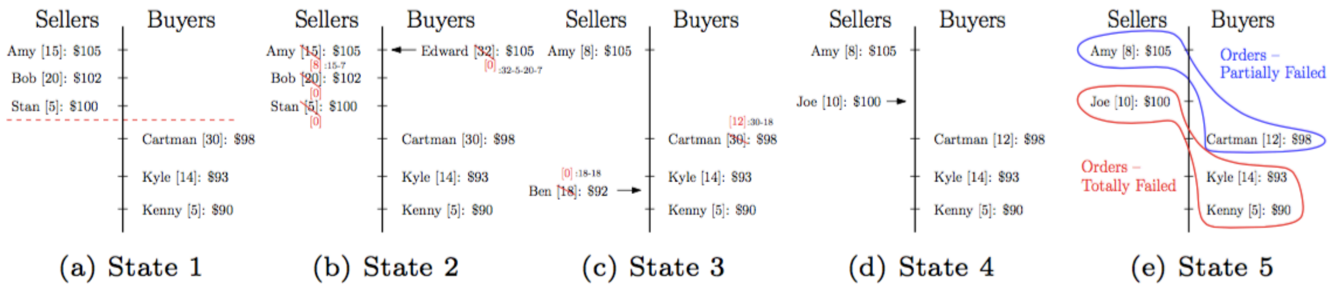


Figure 3: Five subfigures respectively correspond to five different states.

Both Partially Failed orders and Totally Failed orders will be remained in order list and expect in the time slot they can be finished.
However, if these orders can never be finished -- after 1000 time slot, they still are remained in the market, we then say these orders should be canceled.
Thus, we calculate the **cancel rate** by: # of orders remained in market / # of all launched orders.
The Classic Market's cancel rate statistic graph is plotted in Figure 4:

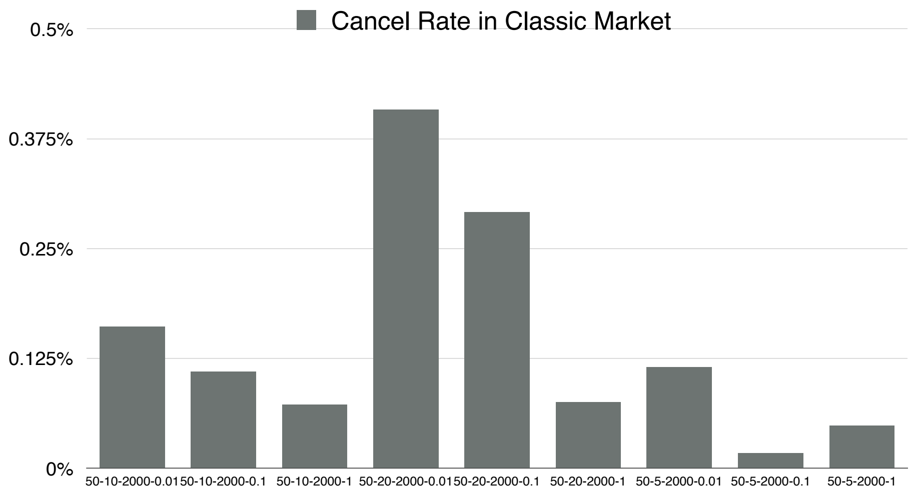


Figure 4: The transaction orders' cancel rate in classic market. The x-axis: **T-R-N-sig**.

3. Comparison Between Bancor and Classic's Cancel Rate

By comparing the transactions' cancel rate between Bancor Market and Classic Market, the result is shown in Figure 5.

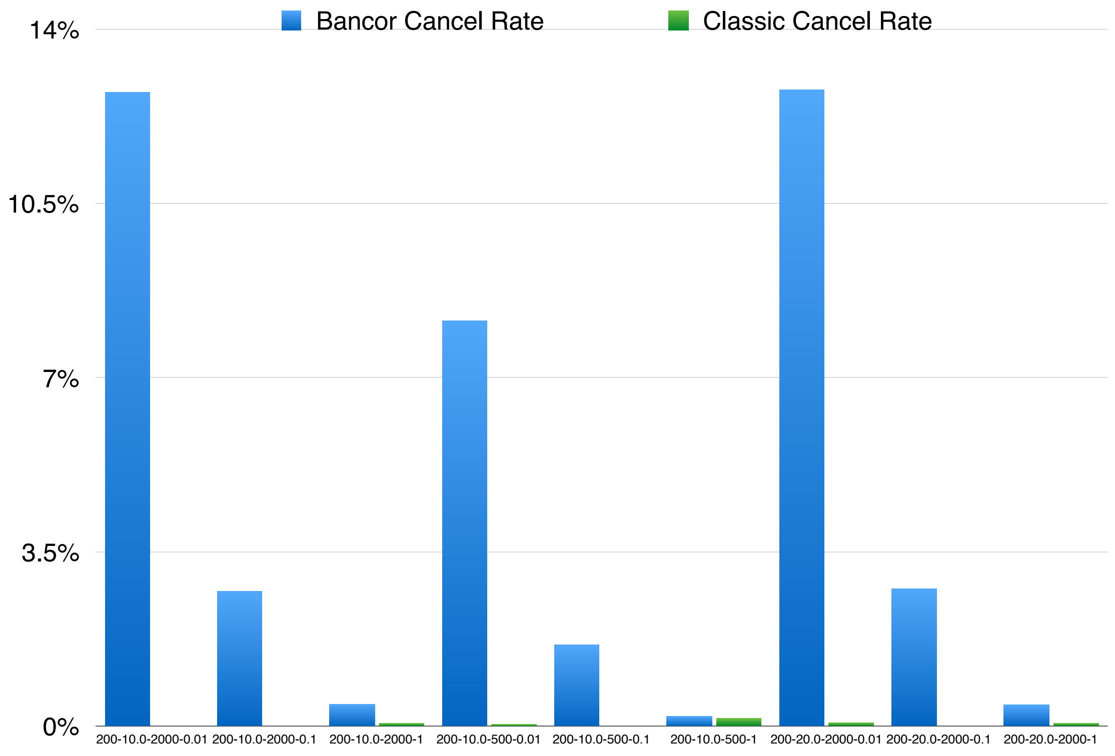


Figure 5: The comparison of cancel rate between Bancor Market and Classic Market. The x-axis: **T-R-N-sig**.

The review of Cancel Rate:
In Bancor:
Why: The price of smart token cannot meet customer's valuation in order.
Cal: cancel rate = # of all canceled orders / # of all launched orders
In Classic:
Why: Customer's order cannot be finished before the end of simulation.
Cal: cancel rate = # of orders unfinished / # of all launched orders.

By Figure 5, we can know the cancel rate in Bancor Market is much higher than in Classic Market.

4. Summary

1. By Figure 1, we know, if customers keep investing money in Bancor Market (never run out of reserve tokens), the market craze in time epoch can lead the price of smart token bouncing around and finally to extremely high.

2. By Figure 2, it can be viewed that the order's cancel rate in Bancor Market can be quite high -- reaching beyond 10% in several parameter settings, especially when customers' valuations are made tightly, i.e., sigma is small.

3. The low classic cancel rate in Figure 4 indicates the "co-incidence of double wants" might not be a problem in Classic Market.

4. By Figure 5, we know the cancel rate in Bancor Market is always much higher than in Classic Market.

The code as well as code's tutorial are presented on:
https://github.com/Ohyoukillkenny/Bancor-Simulator

Unfortunately, up to now, what we have fully discussed in Bancor market is based on **limited order**.
Also, our evaluating metrics might not be representative enough.
Now, I am still working on modifying our simulating model in order to make it feasible in more general cases.