

# Wrapp Master Thesis Task Report

Ziwei Chen

To analyze the contribution of different order strategies, I separate the task into three steps:

1. The Cache.py script caches all logs from page 0 to 3888 to my local disk.
2. Import useful data into my data structure, and handle “error” logs in ReadLog.py

There are mainly two types of “errors” I found in the logs, 1) in the first tens of log pages the “campaigns\_for” log events don't contain “receiver\_id” and “sender\_id”, since an IP can be shared by many users, I cannot rely on IP to match the “sender\_id” of “campaign\_for” with “user” of “reserve\_gift”, I simply ignore this sort of error logs. 2) another type of error is might be a typo from “campaigns” to “campaigns:” in JSON data string, I treat both of them as “campaigns”

I don't store timestamp because the log is already sorted by time, and the linked “campaign\_for” and “reserve\_gift” is restricted by number of logs between them in my algorithm but not time interval, I think the advantage is to reduce memory cost.

My data structure is a list contains two types of tuples, they are (“sender\_id”, “campaigns”, “strategy”) and (“user”, “campaign”)

3. To process the data (Process.py), I go through my log list. If it is a “reserve\_gift” event, I check from the current index - 1 to current index - 5000 (I choose the restrict as 5000 logs because I totally imported 384670 timestamped through 40 days, and I assume two related events should happed within 12 hours), till find out the nearest “campaign\_for” event contains the same “sender\_id” as “user”, also has the “campaign” in the “campaigns” list. Then I record the applied “strategy” into another list and break. I decide to match events in this way because I noticed that in Wrapp webpages it is not allowed to open a series of friends, and it is also not common that a user open several Wrapp homepage or use different clients at the same moment. Therefore I think the nearest log matches the user id and contains the campaign most probably is the “parent” of the “reserve\_gift” event.

Apart from the contributed strategies list, I also retrieve all strategies appeared in the log list into a total strategies list. Then I count the recorded times of each strategies in both list and obtained the devision as contribution rate.

The evaluation result is shown in next page:

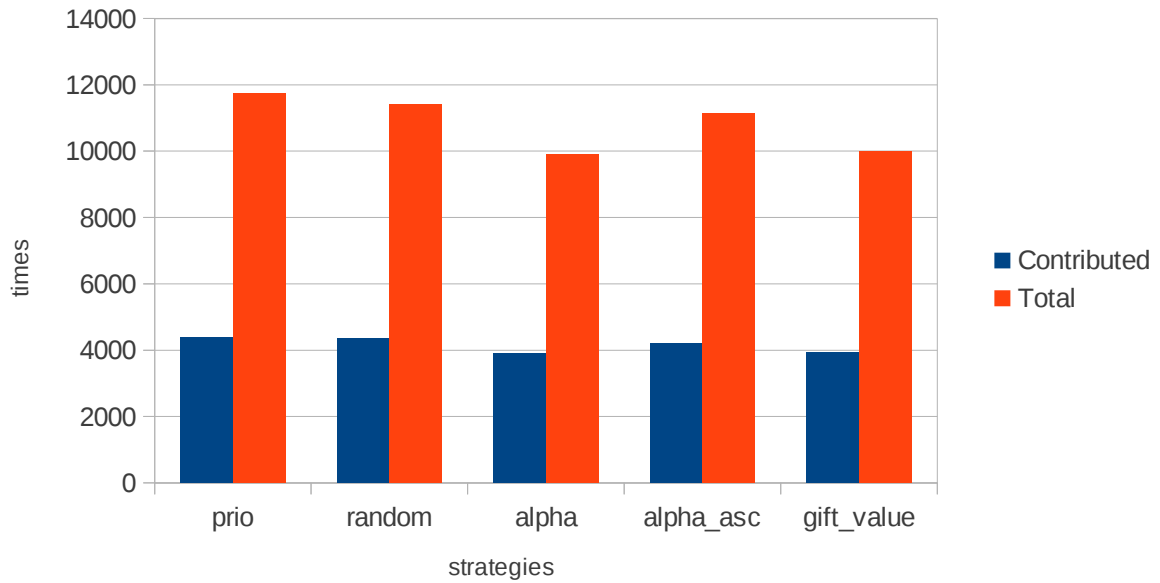


Fig 1 : Strategies contributed and total record times distribution

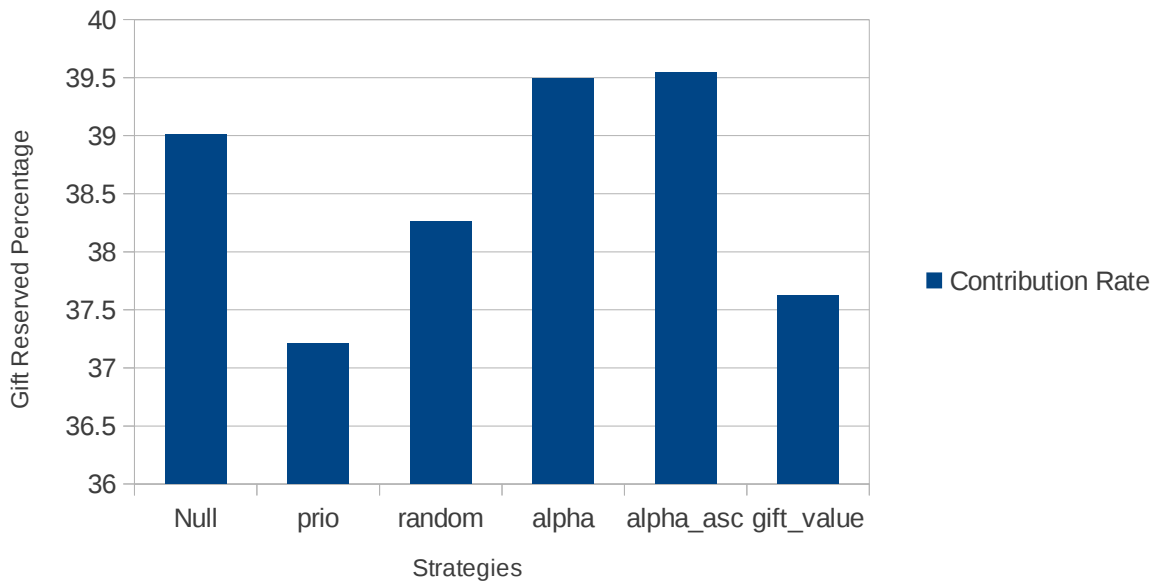


Fig 2 : Strategies contribution rate

In Fig 1, the “no strategy” is not included because the numbers are much greater than events have strategies (total: 221557, contributed:86434). Moreover, those strategies are very close and it is hard to say which is best, because I also tried different size of logs (3500 pages, 3000 pages), and the ranking is different in both Fig 1 and Fig 2.

However, the task asks for a result, I have to say **alpha\_asc strategy** has the best contribution rate and narrowly wins this game.