Chapter 4

Statistics-Based Approach

In a complex system, statistics of different metrics are sometimes not independent. For instance, according to the U.S. Department of Transportation [1], the vehicle miles of travel (VMT) and gross domestic product (GDP) from year 1936 to 2011 are strongly correlated, although they exhibit information in two different areas (transportation and economics).

This motivates us to think about a new approach: rather than collecting statistics of a property directly, we could resort to collecting statistics of other properties that are easier to measure. As long as the correlation between the properties can be established, we could then estimate one property from another.

4.1 Explore the statistics

The goal of this experiment is to show the feasibility of this approach. We also want to know which of the metrics is the most useful for guesstimation.

A simple tool is created to help us obtain statistics of BitTorrent swarms. It has two basic functionalities:

- Download a piece of file and then quit
- Record statistics of our interest when quitting

We are only interested in the statistics when downloading a first piece instead of more pieces, otherwise it would be too expensive (in terms of bandwidth usage) to check the swarm size in this way.

In order to do this, we make use of an open-source library called libtorrent, [2] which is a C++ implementation of BitTorrent. [3]

One of its useful feature is that its internal statistics is accessible from outside, which is helpful for performance tuning and troubleshooting. It also provides some

[1] `https://www.fhwa.dot.gov/policy/otps/pubs/vmt_gdp/`

[2] `http://libtorrent.org/`

[3] The code can be found here: `https://github.com/arvidn/libtorrent`

bindings to other languages (including Python, Java, Go, and Node.js), making it easier to use.

The input to our tool are magnet links or torrents, each representing a unique swarm. For each swarm, our tool will generate a JSON file storing the statistics for later analysis.

The statistics include the following:

- Number of currently connected peers
- Download time of the first piece
- Total number of bytes sent and received by the session
- bittorrent message counters (These counters are incremented every time a message of the corresponding type is received from or sent to a bittorrent peer.)

Add more explanation about these messages

The messages include:

- The **HANDSHAKE** message
- The **CHOKE** message
- The **INTERESTED** message
- The **HAVE** message
- The **BITFIELD** message
- The **CANCEL** message

Some test swarms are chosen as the input for the experiment: Ubuntu [4] and Fedora [5] images.

We choose them because these Linux distribution images are actively downloaded by users and different versions have different popularity. This ensures the variety of the input. In addition, they are all free software so one does not need to worry about legal issues when reproducing the experiment.

4.1.1 Graphs of obtained data

Figure 4.1 shows the swarm sizes of 147 torrents. The data is collected repeatedly for 10 times. We present the data using box plot and further sort the values in ascending order, according to the mean value of each measurement.

From this figure we have these observations:

[4] `http://torrent.ubuntu.com:6969/`
[5] `https://torrent.fedoraproject.org/`

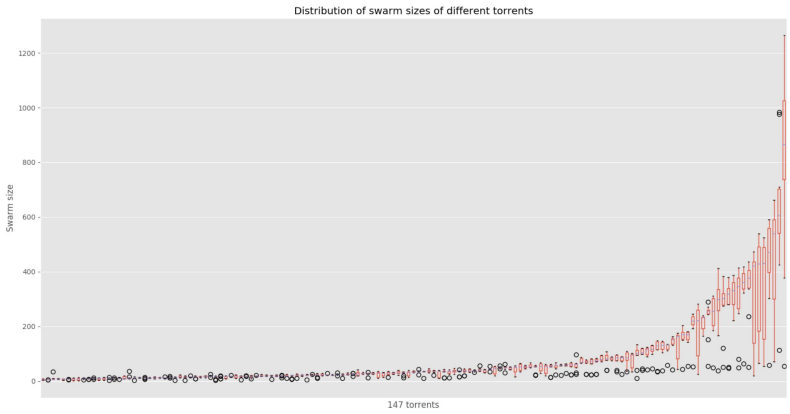


Figure 4.1: Swarm sizes of 147 torrents (measured 10 times for each)

- In general, the swarm sizes of the torrents follows a power law distribution. Most of the torrents have a small swarm size while few of them are quite large.

- When the swarm size is small, the deviation of each measurement is also small. In other words, small swarms tend to remain their sizes. However, large swarms change their size drastically in different measurements.

The first metric we look into the number of connected peers by the time a first piece is downloaded.

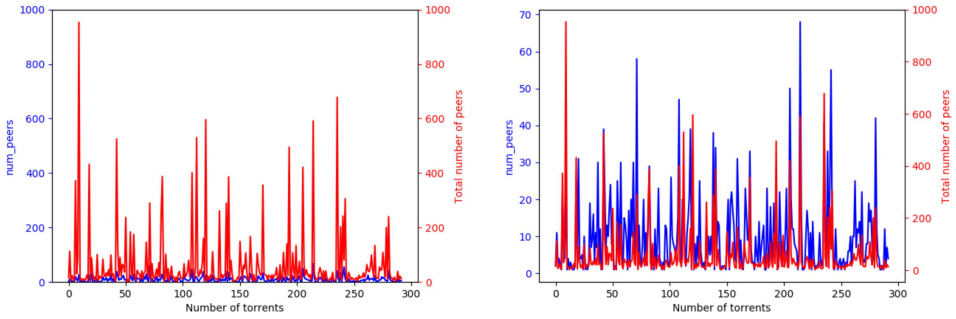
We plot the raw data in the same figure, as shown in Figure 4.2a. It can be observed that the connected peers (in blue) is only a small fraction of the entire set of peers (in red). However, the correlation of the two metrics are not clearly visualized due to their different order of magnitude. Thus, we scale the number of connected peers up in Figure 4.2b.

Unfortunately the relationship is still not visually clear. Therefore, we further sort the swarm size in ascending order, and the corresponding number of currently connected peers is rearranged accordingly. The sorted graphs are Figure 4.3a and Figure 4.3b.

Figure 4.3a simply exhibits the same conclusion as found in Figure 4.2a but in a clearer way. Figure 4.3b now clearly shows us the correlation between the two datasets. The swarm size of the sampled torrents goes up exponentially, while the number of currently connected peers has a similar pattern but with a strong fluctuation when the number of torrents goes up.

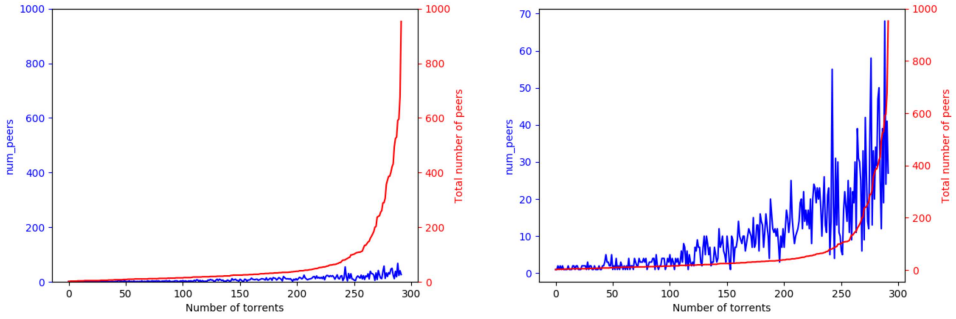
Such a pattern can not only be observed on the number of currently connected peers, but also other metrics. We select some of them to exhibit here:

The next metric we look into is the download time of the first piece.



(a) unscaled (b) rescaled

Figure 4.2: Number of currently connected peers and swarm size (unsorted)



(a) unscaled (b) rescaled

Figure 4.3: Number of currently connected peers and swarm size (sorted)

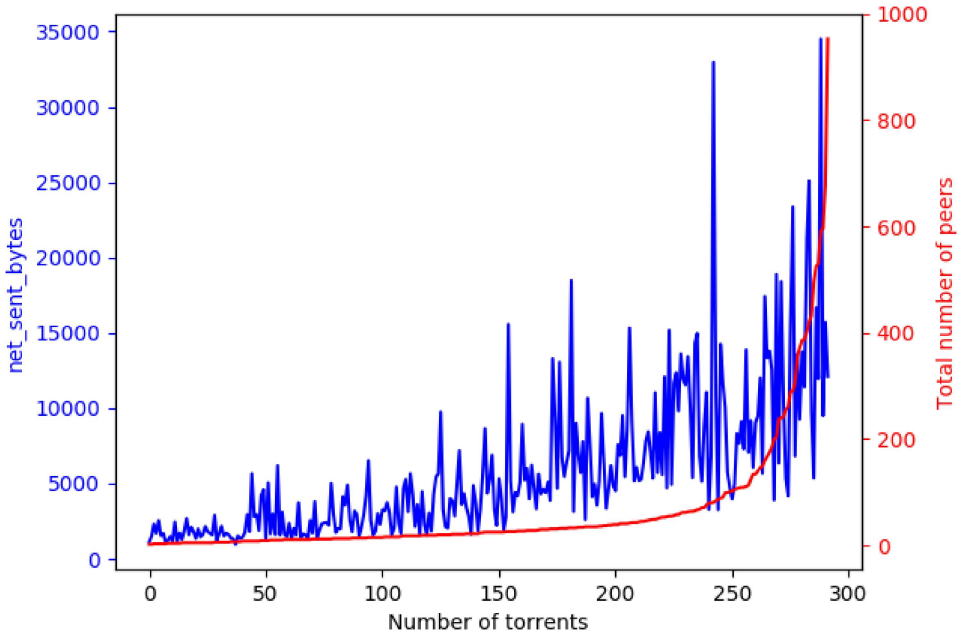


Figure 4.4: Total sent bytes and the swarm size

We can see from Figure 4.6 that in general the download time of the first piece has no evident relationship with swarm size except for some noticeable peaks corresponding to small swarm sizes.

4.1.2 Quantitative Analysis

The graphs has given us an intuition of how the statistics are correlated. We could also presume that Figure 4.4 shows a better correlation that Figure 4.5 because of less fluctuation. In order to prove this we need a quantitative way of describing the correlation.

Three types of correlations are commonly measured for statistical analysis. They are Pearson correlation, Kendall correlation, and Spearman correlation.

Pearson correlation

Suppose we have two datasets $\{x_1, x_2, ..., x_n\}$ and $\{y_1, y_2, ..., y_n\}$, each of them containing n values.

The Pearson's correlation coefficient, conventionally denoted as r, can be calculated by the following formula:

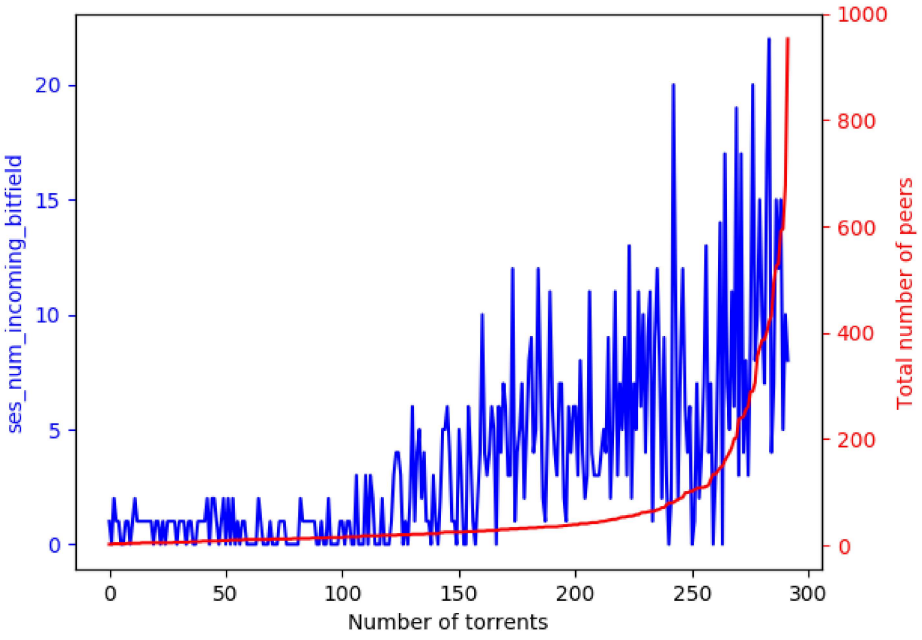


Figure 4.5: Number of incoming bitfield messages and the swarm size

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

where:

- x_i and y_i are the single values in the two datasets with index i
- $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ is the mean of the dataset $\{x_1, x_2, ..., x_n\}$; the same applies to $\bar{y}$

The value of r is always between -1 and $+1$. The sign of the value denotes whether the two datasets are positively or negatively correlated, while the absolute value of it represents the extent to which they are correlated.

Spearman correlation

The Spearman correlation coefficient, denoted as r_s, is defined as the Pearson correlation coefficient between the rank of the variables.

$$r_s = \rho_{rg_X, rg_Y} = \frac{cov(rg_X, rg_Y)}{\sigma_{rg_X}\sigma_{rg_Y}}$$

where:

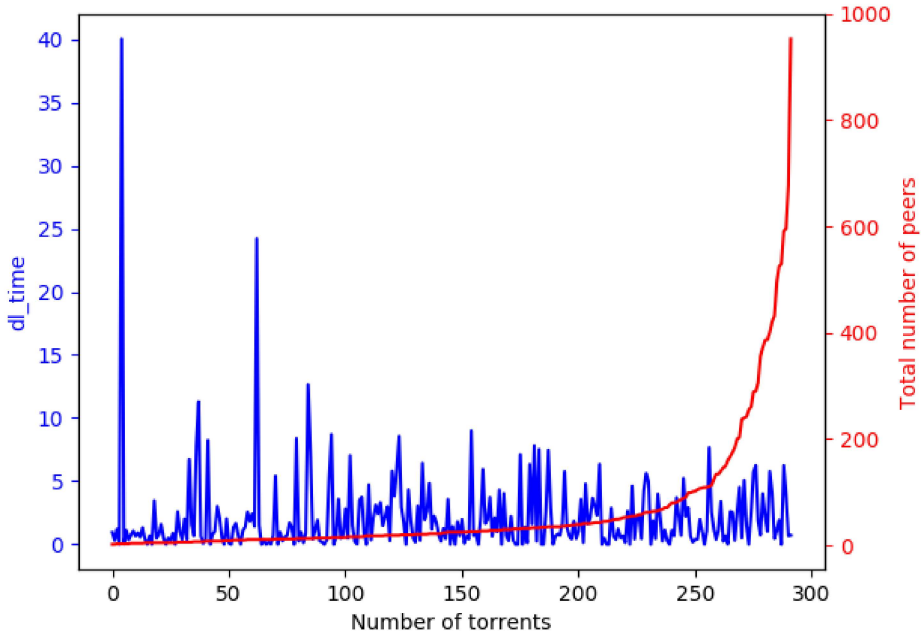


Figure 4.6: Download time of a first piece and the swarm size

- ρ is the Pearson correlation coefficient of the ranked variables
- $cov(rg_X, rg_Y)$ is the covariance of the rank variables
- σ_{rg_X} and σ_{rg_Y} are the standard deviations of the rank variables

Kendall correlation

Any pair of observations (x_i, y_i) and (x_j, y_j) $(i \neq j)$ are said to be *concordant* if $x_i > x_j$ and $y_i > y_j$, or if $x_i < x_j$ and $y_i < y_j$. They are said to be *discordant* when $x_i > x_j$ and $y_i < y_j$, or if $x_i < x_j$ and $y_i > y_j$. If $x_i = x_j$ or $y_i = y_j$, they are neither concordant nor discordant.

The Kendall τ is defined by the following equation:

$$\tau = \frac{(number of concordant pairs) - (number of discordant pairs)}{n(n-1)/2}$$

We calculate these correlation coefficients between different metrics and swarm size. We also plot the data of Table 4.1 in Figure 4.7.

From Figure 4.7 we can clearly see the correlation coefficients of different metrics with the swarm size varies a lot. Thus, we divide them into the following categories:

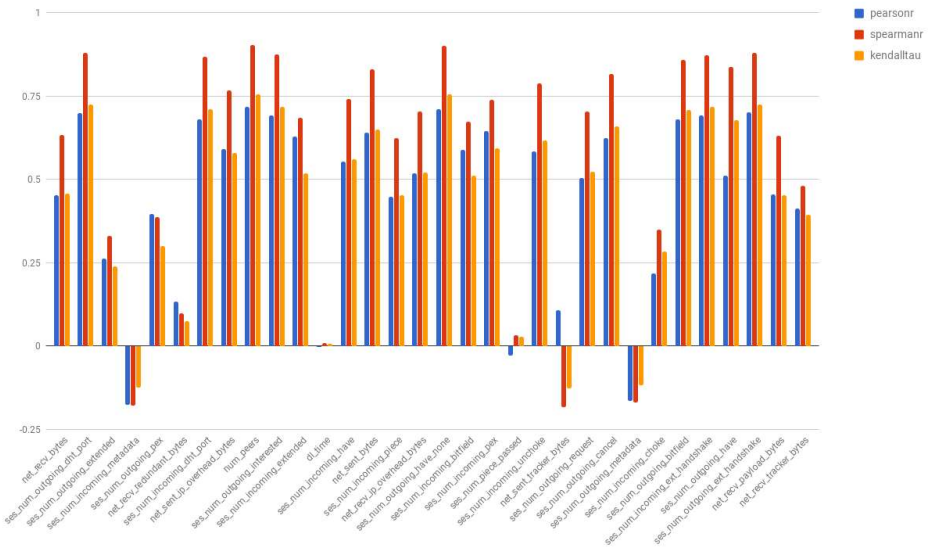


Figure 4.7: Visualization of the correlation coefficients

4.2 Leverage correlation for estimation

What is the reason of dividing them to categories?

We divide the swarms into three categories: *small*, *medium*, and *large*. [6]

4.2.1 Guess from individual metrics

If two variables X and Y are strongly correlated, they are likely to share common characteristics. For example, a *large* X will probably corresponds to a *large* Y, and vice versa.

In our case, we can infer the sizes of swarms from the observed metrics.

We sample a list of metrics $\{m_1, m_2, ..., m_n\}$ and rescale them to the range $[0, 100]$.

If the rescaled value is less than 25, we think that the corresponding swarm to be small; If the rescaled value is between 25 and 75, we think that the corresponding swarm to be of medium size; It the rescaled value is more than 75, we think that the corresponding swarm to be large.

In order to evaluate how well this algorithm works, the rescaling algorithms is applied to the measured swarm sizes. We the count how many estimation results are correct using this as the baseline.

We repeat the experiment for 10 times, and each time we use 11 different metrics to infer the swarm size. A random guess is also generated to compare against the

[6] The concept of large, medium, and small are relative. A swarm size of 100 is a large one compared to 10, but a small one compared to 1000.

Algorithm 1 The swarm size estimation algorithm

```
procedure ESTIMATE_SWARM_SIZE(rescaled_statistics)
    if rescaled_statistics < 25 then
        return Small
    else
        if rescaled_statistics < 75 then
            return Medium
        else
            return Large
```

real swarm size.

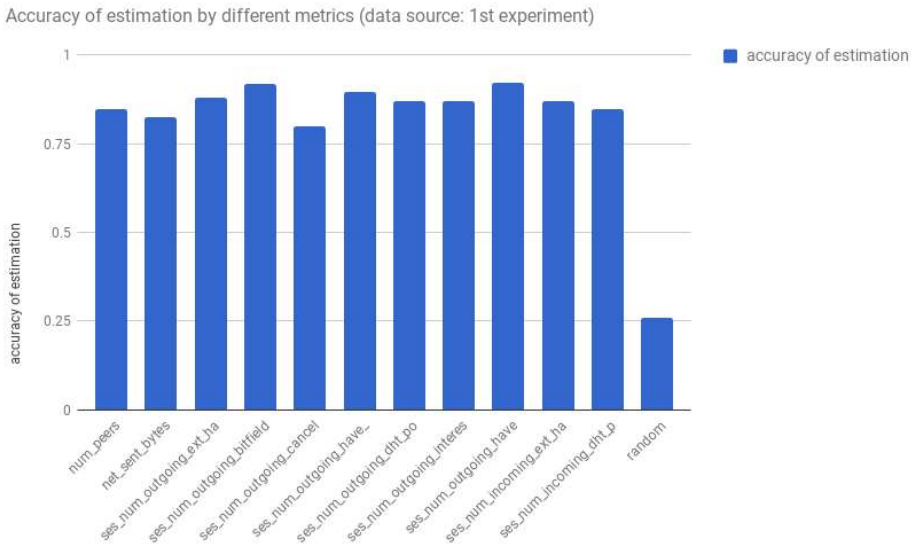


Figure 4.8: Accuracy of estimation by different metrics

Figure 4.8 is the result of one experiment. It shows that:

- Most metrics will have an accuracy of more than 80%
- If we apply this algorithm to a set of random values, the accuracy is around 25%.

The result of all experiments is shown in Figure 4.9.

We see that even in the worst case (Experiment 5) the algorithm works much better than random guess.

[7]

[7]Practical issues: In order to deploy this method, we need a large number of sample torrents to know the inference from raw data to swarm sizes. Calibration may also be required.

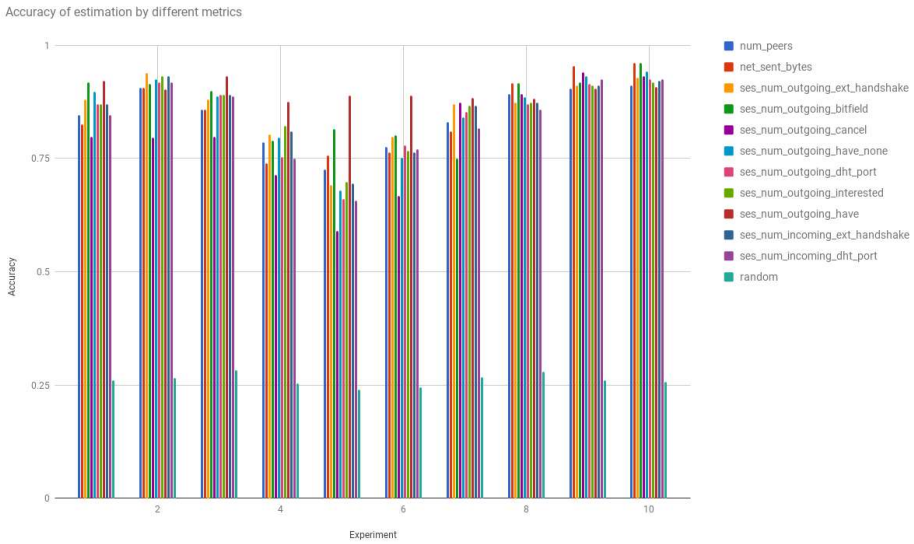


Figure 4.9: Accuracy of estimation by different metrics (10 experiments)

4.2.2 Voting-based approach

We have some voters and each of them have an opinion (i.e. their own estimation) on the size of the swarm. The opinion of the majority is thought of as the swarm size.

The voters are chosen based on their correlation to swarm size. Initially, we choose the metrics that have a strong correlation (See table XX) to swarm size to be the voters.

If voters cannot decide what is the size of a certain swarm because of a tie in the voting result, the least significant voter is removed from the list of voters and the remaining voters vote again. This process continues until a final decision has been made.

Algorithm 2 The voting based swarm size estimation algorithm

```
procedure VOTE(voters)
    for voter ∈ voters do
        list.append(voter.estimate_swarm_size)
    if mode(list) then
        return mode(list)
    else
        return vote(voters.remove_last)
```

We compare the accuracy of estimation with a varying number of voters.

We can see from this chart that:

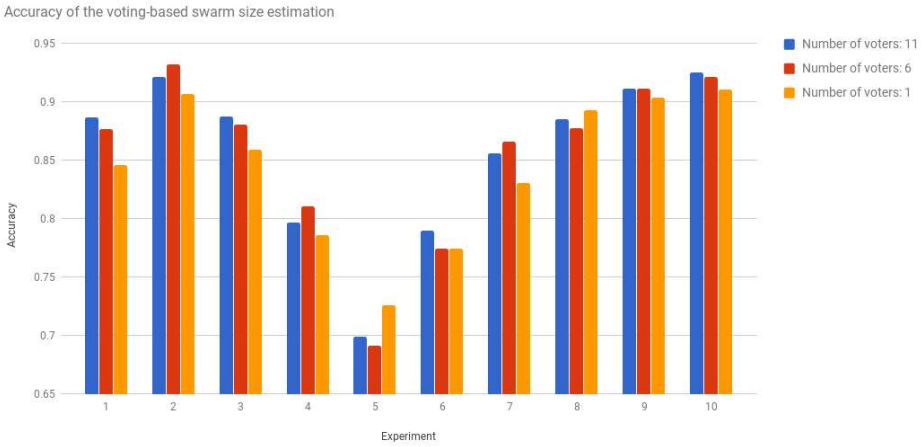


Figure 4.10: Accuracy of voting-based estimation

- More voters will slightly improve the accuracy of estimation by a single metric (we chose num_peers here)
- A large number of voters does not necessarily guarantee a better estimation result

Table 4.1: The correlation coefficient between different metrics and swarm size

name of metrics	pearsonr	spearmanr	kendalltau
net_recv_bytes	0.4526328216	0.6332575134	0.4578871802
ses_num_outgoing_dht_port	0.6986236937	0.8802570577	0.7253182979
ses_num_outgoing_extended	0.2626744685	0.3307728886	0.2397359438
ses_num_incoming_metadata	-0.1771631949	-0.1798132502	-0.1251643986
ses_num_outgoing_pex	0.3964393394	0.3857711987	0.2997454364
net_recv_redundant_bytes	0.132350704	0.09770901602	0.07510976366
ses_num_incoming_dht_port	0.6808422574	0.8678267889	0.711184031
net_sent_ip_overhead_bytes	0.591005004	0.7657741852	0.5798834732
num_peers	0.7166492826	0.9023807213	0.7549144866
ses_num_outgoing_interested	0.6911713704	0.8735221765	0.7177388341
ses_num_incoming_extended	0.6273128646	0.6854177252	0.5187845719
dl_time	-0.003028659706	0.008031282346	0.005800854425
ses_num_incoming_have	0.5527842663	0.7407580796	0.560537692
net_sent_bytes	0.6403528497	0.8311260689	0.6491815874
ses_num_incoming_piece	0.4473741514	0.6237682895	0.4518311691
net_recv_ip_overhead_bytes	0.5184513827	0.7040147508	0.5193734675
ses_num_outgoing_have_none	0.7113675823	0.9016118762	0.7553611303
ses_num_incoming_bitfield	0.5892387847	0.6727475351	0.5108201316
ses_num_incoming_pex	0.6446127691	0.7388309133	0.592987013
ses_num_piece_passed	-0.02930296135	0.03271200907	0.02698558043
ses_num_incoming_unchoke	0.5840626368	0.7874070614	0.6170817143
net_sent_tracker_bytes	0.1081686955	-0.1837134923	-0.1267368397
ses_num_outgoing_request	0.5047942801	0.7032072207	0.5217074923
ses_num_outgoing_cancel	0.6233354535	0.8152422867	0.6583026061
ses_num_outgoing_metadata	-0.1648271538	-0.1707369191	-0.1186882375
ses_num_incoming_choke	0.2171426974	0.3481792088	0.2838047977
ses_num_outgoing_bitfield	0.6799232645	0.8584115588	0.7090649408
ses_num_incoming_ext_handshake	0.6912972726	0.8732157467	0.717362905
ses_num_outgoing_have	0.5103046396	0.8359766119	0.6776197444
ses_num_outgoing_ext_handshake	0.7000594065	0.8791635098	0.7242431541
net_recv_payload_bytes	0.4554698315	0.6300260259	0.4524761851
net_recv_tracker_bytes	0.411920382	0.480568137	0.39332504

Table 4.2: Category of correlation with swarm size

Name of metrics	Correlation with swarm size
num_peers	Strong
ses_num_outgoing_have_none	
ses_num_outgoing_dht_port	
ses_num_outgoing_ext_handshake	
ses_num_outgoing_interested	
ses_num_incoming_ext_handshake	
ses_num_incoming_dht_port	
ses_num_outgoing_bitfield	
ses_num_outgoing_have	
net_sent_bytes	
ses_num_outgoing_cancel	
ses_num_incoming_unchoke	Medium
net_sent_ip_overhead_bytes	
ses_num_incoming_have	
ses_num_incoming_pex	
net_recv_ip_overhead_bytes	
ses_num_outgoing_request	
ses_num_incoming_extended	
ses_num_incoming_bitfield	
net_recv_bytes	
ses_num_outgoing_extended	Low
ses_num_outgoing_pex	
ses_num_incoming_choke	
net_recv_tracker_bytes	
net_recv_payload_bytes	
ses_num_incoming_piece	
net_recv_redundant_bytes	
ses_num_piece_passed	None
dl_time	
ses_num_outgoing_metadata	Negative
ses_num_incoming_metadata	
net_sent_tracker_bytes	Uncategorized