# Peer to Peer File System Clients

Kojin Oshiba & Roman Sigalov

February 12, 2016

## 1 Analysis

Provide your answers based on test runs with the clients you programmed. When asked for comparative performance results, provide some evidence (e.g., simple statistics) for the results you are reporting/describing.

(a) For the standard client explain what assumptions or decisions you had to make beyond those specified in Chapter 5.

For this part we made 4 main assumptions:

- We set total number of slots for our client equal to 4
- We defined the rarest piece the following way: piece that has the lowest number of owners, i.e. peers that have that particular peer. We designed the client in such way that out of all pieces that it need to download, first it tries to get the rarest ones.
- We implemented reciprocation in the following way: each round we calculate the number of blocks that each peer who requested something from us uploaded to our client in the previous two periods (or the largest number of available rounds) and take top three peers based on this value.
- Optimistic unchoking was implemeted the following way: each third period (i.e. $0,3,6,\ldots$) after deciding on which players to unchoke based on reciprocation the client chooses random peer from his neighborhood including peer that didn't requested anything and allocates optimistically unchoked slot to this peer. This peer remains unchoked for this and two subsequent rounds.

(b) Write a concise summary of the strategies you used for the tournament client, and why you chose them.

Our tournament client is based on BitTyrant. However, we have made three modifications as specified below.

1. When the client has 95% of the pieces it wants to collect, it sets a cap lower than the maximum bandwidth to its upload capacity. This is because, it doesn't need to be as nice to others as before in order to accumulate its "good reputation" to higher its probability of getting unchoked since it's about to end its piece collecting. Instead, it wants to be mean to others by not uploading its pieces to others and slow down competitors piece collection.

2. When the client has $x$% of the pieces needed to be collected, we allocate $50 - \frac{x}{2}$% of its slots to optimistic unchoking. This is because, initially, it wants to believe more in luck when unchoking people, whereas once it starts to get a lot of information, it can base more of their unchoking based on the existing information.

3. The client breaks the assumption of rarest first. Rarest first is effective if we assume the other clients are randomly choosing the pieces to request. However, when many players are BitTyrrant or reference client like (which will likely be the case in the competition), rarest first isn't the best strategy because everyone will be trying to download rarest pieces and thus there will be a competition. So, in our tournament client, we first calculate the rarest pieces just like other clients do. However, we eliminate them from our request list of pieces. Instead, we run another calculation for the remaining pieces to find the rarest ones among them. In this way, we are able to avoid competition but still request moderately rare pieces.

4. The client mimics the Dummy client in rounds $1, 4, 7, \ldots$ and assigns all bandwidth to some random peer. We randomly tried this and it seems to work well.

(c) Outperforming the standard client:

i. How does the BitTyrant client do in a population of standard clients?

ii. How does the Tourney client do in a population of standard clients?

iii. How does the PropShare client do in a population of standard clients? Look at the relative ranking of the clients and the percentage improvement (or impair- ment) in the number of rounds it takes the client to get the complete file.

i. Performance of BitTyrant in a population of standard reference clients is presented in tables 2.1 and 2.2. Each column represents a client: **Bold** number represent mean and standard deviation of rounds to completion for BitTyrant and ordinary numbers number represent mean and standard deviation of rounds to completion for reference client. In Table 2.1 there are many iterations of the simulations with the same settings. As we can see, Tyrant performed better than the average and it is frequently appers in the top of the ranking which suggest that on average in these settings (parmeters of simulation) BitTyrant is able to be faster than the reference client. When we increase number of iterations of the same settings to 500 (in Table 2.2 the same pattern persists: BiTyrant still performs better than the average. The same picture is in the second column where we changed numbers of pieces and blocks and *max_bw*.

   However, if we change the parameters further, for example, by making more pieces and blocks which corresponds to moving from left to right along with table 2.2 the performance of ButTyrant client decreases significantly and it is not able be significantly better than the average and rather it becomes just the average reference client.

ii. Performance of Tourney client in a population of standard reference clients is presented in tables 2.6 and 2.7. As we can see in Table 2.6, Tourney client out performs very well when simulated with solely BitTorrent clients. However, things didn't turn out that easy. in Table 2.7, we can see that the performance varies depending on how we set the variables. Particularly, by comparing the third one and the fourth one, we know that the decrease in max bw result in lower performance. The competition will be using max bw = 64, so this shouldn't cause a problem. Comparing the second and the fourth, we can also see that the increase in blocks slightly decreases the performance. This might be a problem since blocks=32 in the competition, but the negative effect shouldn't be as significant as the decrease in max bw.

iii. Performance of PropShare client in a population of standard reference clients is presented in Table 2.4 and 2.5. As we can see in Table 2.4 PropShare client underperfomes compared to reference client and the deviation is quite large. However, only in two interations it was able to make it to the top and otherwise it was in the second half of the client rating for each iteration. When we try different settings in Table 2.5 the same situation persists. 500 iterations with the same parameters shows the same result and changing parameters to bigger number of pieces and blocks significantly dcreases performance of PropShare client suggesting that it using it in the population of reference client seems to be irrational.

(d) Overall performance of populations:

i. How does a population of only BitTyrant clients perform? What about a population of only Tourney clients?

ii. How does a population of only PropShare clients do?

Look at the time it takes to get the file out to all clients (i.e., when does the last client complete downloading the whole file), as well as the average download time for the individual clients.

The results for simulation for single-client simulation are shows in Table 2.3. We can see a very interesting situation. It seems like the client that is used in population doesn't matter for the total welfare which we can define as a maximum completion round. In each case three of our clients: reference, BitTyrant and PropShare give the same total welfare and this picture persists even if we change parameters. This conclusion surptising, but logic, though. When we have some way strategy that is related to tit-for-tat we are expected to fell into an equilibrium with a slow download rate, because we should protect the system from freriding which makes it slow and doesn't give an opportuinity to increase speed.

Which is more inportant is that Dummy client which simply chooses random peer to unchoke outperforms each client in term of "total welfare". The reason is that dummy represents the state of total cooperation: we always ready to give each peer the total bandwidth and it doesn't depend on previous behavior of each agent. Thereby we can conlude that this state can be closer to pareto-optimum compared to other strategies that we tried to implement i this assignment.

(e) Write a paragraph about what you learned from these exercises about BitTorrent, game theory, and programming strategic clients? (We arenâĂŹt looking for any particular answers here, but are looking for evidence of real reflection.)

The most important thing that we learned is the fact that the theory doesn't work easily in practice. For example, our BitTyrant should supposed to outperform other clients theoretically. However, the result was above average and it didn't perform as well as we imagined. This was the same for our tournament client. It performed really well under certain conditions but not at all in others. In short, you never know whether a certain feature is critical until you actually test it. Even so, we still found game theory to be an amazing economic model, being able to draw out ways to implement and analyze such a complicated system like BitTorrent. We always thought "how can actual phenomena in the society can be as simple as games in game theory" but it captures the behaviors of clients wonderfully accurately.

## 2 Theory

(a) State three ways in which the peer-to-peer file sharing game of the BitTorrent network is different from a repeated PrisonerâĂŹs dilemma.

The first difference is that the P2P game has a lot richer set of action: you can not only cooperate and defect but also to choose strategically reveal downloaded pieces, adjust upload rate to your peers, decide how many peers can download from you for each give round.

The second difference is that there more than two player. This makes computing the equilibrium and designing the winning strategy far more complicated because your download speed depends not only on your actions and on actions of your peer, but it also depends on behavior of other neighbors in the swarm.

The Third difference is that the game can end not simultaneously for any player. Rather there can be a situation where some peer $j$ will not longer be interested in any cooperation with peer $i$ because peer $i$ doesn't have unique pieces that peer $j$ doesn't have.

Table 2.1: BitTyrant client (**bold**) vs 10 BitTorrent clients. Av. completion rounds (st. dev)

| | | | | | |
|---|---|---|---|---|---|
| 25.64 (1.28) | 25.28 (1.31) | 25.28 (1.31) | 25.26 (1.05) | 25.5 (1.02) | 25.52 (1.06) |
| **25.66 (1.16)** | 25.32 (1.01) | 25.32 (1.01) | 25.54 (1.06) | 25.68 (1.09) | 25.64 (1.21) |
| 25.7 (1.10) | 25.36 (1.05) | 25.36 (1.05) | 25.6 (1.02) | **25.72 (1.06)** | **25.72 (1.18)** |
| 25.76 (1.30) | 25.38 (1.32) | 25.38 (1.32) | 25.64 (1.20) | 25.72 (1.06) | 25.74 (1.04) |
| 25.8 (1.02) | 25.46 (1.17) | 25.46 (1.17) | **25.66 (1.09)** | 25.74 (1.15) | 25.76 (1.09) |
| 25.82 (1.32) | 25.48 (1.17) | 25.48 (1.17) | 25.66 (1.16) | 25.8 (1.11) | 25.86 (0.96) |
| 25.82 (1.19) | 25.52 (1.25) | 25.52 (1.25) | 25.68 (1.26) | 25.82 (1.21) | 25.92 (0.98) |
| 25.88 (1.24) | **25.62 (1.04)** | **25.62 (1.04)** | 25.74 (1.18) | 25.82 (1.42) | 26 (1.00) |
| 25.92 (1.02) | 25.62 (1.28) | 25.62 (1.28) | 25.78 (1.19) | 25.88 (1.27) | 26.06 (1.14) |
| 25.94 (1.29) | 25.72 (1.20) | 25.72 (1.20) | 25.84 (1.05) | 25.92 (1.16) | 26.08 (1.00) |
| 26.12 (1.37) | 25.82 (1.16) | 25.82 (1.16) | 25.88 (1.34) | 25.94 (0.99) | 26.14 (1.18) |
| 25.48 (1.15) | 25.22 (1.15) | 25.56 (0.92) | 25.38 (1.28) | 25.42 (0.94) | 25.56 (1.02) |
| 25.52 (1.19) | **25.32 (1.07)** | 25.6 (1.06) | 25.4 (1.02) | 25.44 (1.13) | 25.64 (1.16) |
| 25.56 (0.94) | 25.34 (0.99) | **25.62 (1.25)** | 25.42 (0.94) | 25.44 (1.06) | **25.68 (1.03)** |
| 25.58 (1.25) | 25.34 (1.11) | 25.64 (1.18) | 25.44 (1.13) | 25.5 (1.20) | 25.68 (1.24) |
| 25.68 (1.09) | 25.42 (1.06) | 25.66 (1.12) | **25.46 (1.00)** | **25.56 (1.06)** | 25.78 (1.29) |
| **25.7 (1.06)** | 25.42 (1.04) | 25.66 (1.19) | 25.48 (0.88) | 25.56 (0.85) | 25.78 (1.17) |
| 25.78 (0.99) | 25.48 (0.88) | 25.66 (1.14) | 25.54 (1.20) | 25.58 (1.06) | 25.78 (1.32) |
| 25.8 (1.10) | 25.48 (1.15) | 25.74 (0.93) | 25.54 (1.10) | 25.6 (0.89) | 25.8 (0.85) |
| 25.8 (1.11) | 25.54 (1.15) | 25.78 (1.22) | 25.66 (1.01) | 25.62 (1.07) | 25.86 (1.04) |
| 25.84 (1.05) | 25.68 (1.01) | 25.86 (1.02) | 25.7 (1.06) | 25.7 (0.96) | 25.9 (1.22) |
| 25.88 (1.03) | 25.76 (1.32) | 25.86 (0.98) | 25.74 (1.00) | 25.78 (1.15) | 25.98 (1.19) |

The following simulation parameters were used:
*–num-pieces=20 –blocks-per-piece=10 –min-bw=16 –max-bw=32 –iters=50*

(b) State three ways in which the BitTorrent reference client is different from the tit-for-tat strategy in a repeated PrisonerâĂŹs Dilemma.

(cite:textbook p110) In BitTorrent, agents can not only "decide between sharing and not sharing, but they can also vary how much upload bandwidth to make available to each peer".

"Whether peer j receives any bandwidth from peer i depends not only on peer jâĂŹs actions, but also on the actions of other peers in iâĂŹs neighborhood."

"A peer j may get to the point where it no longer has any pieces that i wants." In Prisoner's Dilemma, agents would want to maximize their utilities endlessly.

(c) Suppose there is a client that is a NE when adopted by every user. Give two reasons why the system of users nevertheless switch over time to use another client.

All clients don't perform completely rationally as in the case of optimistic unchoking. So there is a chance that this randomness triggers some clients to deviate from NE.

Not every player is playing the same number of rounds. Therefore, once someone finishes downloading, they will start to behave in different ways and thus break the equilibrium.

Therefore, people would switch overtime to use other clients.

Table 2.2: BitTyrant client (**bold**) vs 10 BitTorrent clients. Av. completion rounds (st. dev)

| Simulation parameters: | | | | | | |
|---|---|---|---|---|---|---|
| min_bw | 16 | 16 | 16 | 16 | 16 | 16 |
| max_bw | 32 | 64 | 32 | 64 | 64 | 64 |
| pieces | 20 | 64 | 64 | 64 | 128 | 128 |
| blocks | 10 | 16 | 32 | 32 | 32 | 32 |
| iters | 500 | 50 | 50 | 50 | 50 | 50 |
| Simulation results: | | | | | | |
| | 25.5 (1.02) | 63.5 (0.50) | 267.84 (2.96) | 128.02 (0.79) | 255.78 (0.73) | 255.84 (0.81) |
| | 25.51 (1.10) | 63.58 (0.57) | 268.12 (2.85) | 128.06 (0.88) | 255.92 (0.77) | 255.9 (0.85) |
| | **25.6 (1.08)** | **63.7 (0.73)** | 268.18 (2.67) | 128.08 (0.82) | 256 (0.69) | 256.12 (0.77) |
| | 25.632 (1.06) | 63.72 (0.60) | 268.22 (2.94) | 128.1 (0.96) | 256 (0.85) | 256.16 (0.88) |
| | 25.662 (1.12) | 63.72 (0.69) | 268.3 (3.27) | 128.12 (0.86) | 256.02 (0.91) | 256.2 (0.82) |
| | 25.686 (1.06) | 63.74 (0.77) | 268.42 (2.64) | 128.12 (0.86) | **256.22 (0.86)** | 256.26 (0.80) |
| | 25.742 (1.08) | 63.76 (0.59) | 268.44 (3.05) | **128.18 (0.89)** | 256.22 (0.83) | **256.32 (0.84)** |
| | 25.754 (1.18) | 63.8 (0.69) | **268.48 (3.21)** | 128.18 (0.68) | 256.28 (0.80) | 256.34 (0.89) |
| | 25.786 (1.12) | 63.82 (0.55) | 268.52 (2.99) | 128.28 (0.75) | 256.54 (0.90) | 256.5 (0.90) |
| | 25.788 (1.15) | 63.98 (0.71) | 268.56 (3.06) | 128.4 (0.94) | 256.66 (0.91) | 256.54 (0.96) |
| | 25.81 (1.17) | 64.02 (0.65) | 268.66 (3.33) | 128.42 (0.83) | 256.66 (0.86) | 256.58 (0.78) |

Table 2.3: Population performance

| Simulation parameters: | | | | | | | |
|---|---|---|---|---|---|---|---|
| min_bw | 10 | 10 | 16 | 16 | 16 | 16 | 16 |
| max_bw | 20 | 20 | 32 | 64 | 64 | 64 | 64 |
| pieces | 20 | 32 | 32 | 32 | 64 | 64 | 128 |
| blocks | 10 | 16 | 16 | 16 | 16 | 32 | 32 |
| Last client completes download in (rounds): | | | | | | | |
| Reference | 39 | 118 | 65 | 33 | 65 | 129 | 257 |
| Tyrant | 39 | 119 | 64 | 32 | 64 | 129 | 257 |
| PropShare | 38 | 116 | 65 | 33 | 65 | 129 | 257 |
| Tourney | 39 | 120 | 64 | 32 | 62 | 131 | 259 |
| Dummy | 22 | 73 | 30 | 15 | 25 | 63 | 102 |

* Each simulation used 10 clients and 2 seeds

Table 2.4: PropShare client (**bold**) vs 10 BitTorrent clients. Av. completion rounds (st. dev)

| | | | | | |
|---|---|---|---|---|---|
| 25.52 (0.94) | 25.3 (1.06) | 25.52 (1.08) | 25.22 (1.06) | 25.48 (0.88) | 25.4 (0.87) |
| 25.58 (1.10) | 25.46 (1.06) | 25.6 (1.08) | 25.26 (0.98) | 25.56 (1.17) | 25.54 (1.00) |
| 25.62 (1.07) | 25.52 (1.20) | 25.6 (1.02) | 25.4 (1.06) | 25.58 (1.06) | 25.56 (0.92) |
| 25.64 (1.11) | 25.56 (1.12) | 25.66 (1.11) | 25.42 (1.13) | 25.7 (1.02) | 25.66 (0.91) |
| 25.7 (0.96) | 25.6 (1.11) | 25.7 (1.20) | 25.48 (1.22) | 25.7 (1.00) | 25.68 (0.99) |
| 25.7 (1.04) | 25.72 (1.08) | 25.7 (1.17) | 25.52 (1.02) | 25.72 (0.98) | **25.7 (1.10)** |
| 25.7 (1.08) | 25.74 (1.13) | 25.72 (1.27) | 25.6 (1.15) | **25.76 (1.01)** | 25.7 (1.19) |
| 25.7 (1.14) | **25.76 (1.01)** | 25.78 (1.08) | **25.66 (1.11)** | 25.76 (1.09) | 25.74 (1.18) |
| **25.86 (1.17)** | 25.76 (1.16) | **25.8 (1.11)** | 25.66 (1.18) | 25.8 (1.06) | 25.76 (1.19) |
| 25.94 (1.16) | 25.78 (1.03) | 25.86 (1.00) | 25.7 (1.08) | 25.84 (1.10) | 25.86 (1.00) |
| 25.94 (1.10) | 25.8 (1.13) | 25.9 (1.10) | 25.78 (0.99) | 25.96 (1.23) | 25.9 (0.98) |
| 25.4 (0.96) | 25.34 (1.21) | **25.52 (1.12)** | 25.44 (0.94) | 25.54 (1.06) | 25.54 (1.06) |
| 25.46 (1.00) | 25.38 (1.13) | 25.58 (1.10) | 25.5 (1.06) | 25.58 (1.02) | 25.58 (1.02) |
| 25.56 (1.02) | 25.42 (1.20) | 25.6 (0.80) | 25.52 (1.06) | 25.58 (1.10) | 25.58 (1.10) |
| 25.58 (1.06) | 25.46 (0.88) | 25.62 (1.09) | **25.66 (1.01)** | 25.64 (1.16) | 25.64 (1.16) |
| 25.66 (1.34) | 25.52 (1.00) | 25.64 (1.00) | 25.68 (1.03) | **25.68 (0.84)** | **25.68 (0.84)** |
| **25.72 (1.11)** | 25.58 (1.15) | 25.68 (1.07) | 25.68 (1.09) | 25.74 (1.02) | 25.74 (1.02) |
| 25.76 (0.97) | **25.6 (1.13)** | 25.68 (0.97) | 25.7 (1.19) | 25.78 (1.08) | 25.78 (1.08) |
| 25.82 (1.07) | 25.62 (1.20) | 25.72 (1.15) | 25.76 (1.14) | 25.8 (1.17) | 25.8 (1.17) |
| 25.88 (1.24) | 25.66 (1.16) | 25.72 (1.08) | 25.92 (1.13) | 25.88 (1.27) | 25.88 (1.27) |
| 25.96 (1.09) | 25.66 (1.14) | 25.8 (1.28) | 25.92 (1.06) | 25.9 (0.96) | 25.9 (0.96) |
| 26.12 (1.09) | 25.78 (1.15) | 25.98 (1.30) | 25.94 (1.19) | 25.92 (1.20) | 25.92 (1.20) |

The following simulation parameters were used:
*–num-pieces=20 –blocks-per-piece=10 –min-bw=16 –max-bw=32 –iters=50*

Table 2.5: PropShare client (**bold**) vs 10 BitTorrent clients. Av. completion rounds (st. dev)

| Simulation parameters: | | | | | |
|---|---|---|---|---|---|
| min_bw | 16 | 16 | 16 | 16 | 16 |
| max_bw | 32 | 64 | 32 | 64 | 64 |
| pieces | 20 | 64 | 64 | 64 | 128 |
| blocks | 10 | 16 | 32 | 32 | 32 |
| iters | 500 | 50 | 50 | 50 | 50 |
| Simulation results: | | | | | |
| | 25.54 (1.04) | 63.64 (0.62) | 268.3 (2.98) | 127.94 (0.76) | 255.96 (0.75) |
| | 25.6 (1.17) | 63.66 (0.62) | 268.44 (2.94) | 128 (0.85) | 256 (0.82) |
| | 25.7 (1.14) | 63.68 (0.61) | 268.48 (3.26) | 128.02 (0.88) | 256.06 (0.88) |
| | 25.7 (1.17) | 63.7 (0.67) | 268.6 (3.46) | 128.06 (0.79) | 256.24 (0.71) |
| | **25.72 (1.37)** | 63.72 (0.69) | 268.62 (3.38) | 128.14 (0.89) | 256.28 (0.90) |
| | 25.72 (1.31) | 63.74 (0.59) | 268.7 (3.12) | 128.2 (0.87) | 256.28 (0.98) |
| | 25.74 (1.16) | 63.8 (0.75) | 268.76 (3.46) | 128.26 (0.84) | 256.3 (0.83) |
| | 25.74 (1.16) | 63.82 (0.71) | 268.76 (3.43) | 128.32 (0.68) | 256.46 (0.70) |
| | 25.86 (1.25) | 63.92 (0.52) | **269 (3.16)** | 128.44 (1.02) | **256.48 (0.85)** |
| | 25.98 (1.14) | **63.98 (0.71)** | 269.1 (3.45) | 128.44 (0.98) | 256.56 (0.80) |
| | 26 (1.11) | 64.06 (0.79) | 269.12 (3.10) | **128.54 (0.98)** | 256.7 (0.83) |

Table 2.6: Tourney client (**bold**) vs 10 BitTorrent clients. Av. completion rounds (st. dev)

| | | | | | |
|---|---|---|---|---|---|
| **9.46 (1.04)** | **9.58 (0.98)** | **9.12 (1.07)** | **9.3 (1.06)** | **9.4 (1.20)** | **9.1 (1.00)** |
| 9.5 (0.85) | 9.68 (0.90) | 9.28 (0.90) | 9.52 (0.96) | 9.76 (0.93) | 9.32 (0.73) |
| 9.62 (0.91) | 9.76 (0.79) | 9.4 (1.00) | 9.54 (0.78) | 9.76 (0.81) | 9.4 (0.77) |
| 9.74 (0.82) | 9.78 (0.81) | 9.4 (0.87) | 9.76 (0.84) | 9.78 (0.88) | 9.48 (0.90) |
| 9.76 (1.01) | 9.94 (0.81) | 9.42 (0.83) | 9.78 (0.90) | 9.82 (0.91) | 9.6 (0.80) |
| 9.76 (0.84) | 9.96 (0.72) | 9.54 (0.78) | 9.82 (0.95) | 9.82 (0.77) | 9.6 (0.85) |
| 9.78 (0.97) | 10.02 (0.79) | 9.56 (0.78) | 9.94 (0.83) | 10.02 (0.81) | 9.74 (0.80) |
| 10.06 (0.79) | 10.32 (0.76) | 9.88 (0.93) | 9.98 (0.97) | 10.22 (0.88) | 9.86 (0.89) |
| 10.08 (0.91) | 10.38 (0.91) | 9.94 (0.93) | 10.06 (0.81) | 10.26 (0.89) | 9.88 (0.82) |
| 10.16 (0.88) | 10.46 (0.90) | 10 (0.75) | 10.14 (0.82) | 10.66 (0.84) | 10.02 (0.84) |
| 10.26 (0.89) | 10.54 (0.81) | 10.08 (0.87) | 10.22 (0.92) | 10.66 (0.89) | 10.1 (0.83) |

The following simulation parameters were used:
*–min-bw=16 –max-bw=32 –num-pieces=20 –blocks-per-piece=10 –max-round=1000 –iters=50*

Table 2.7: Tourney client (**bold**) vs 10 BitTorrent clients. Av. completion rounds (st. dev)

| Simulation parameters: | | | | |
|---|---|---|---|---|
| min_bw | 16 | 16 | 16 | 16 |
| max_bw | 32 | 64 | 32 | 64 |
| pieces | 20 | 64 | 64 | 64 |
| blocks | 10 | 16 | 32 | 32 |
| iters | 50 | 50 | 50 | 50 |
| Simulation results: | | | | |
| | 22.7 (1.62) | **41.56 (2.33)** | 232.9 (10.11) | 96.14 (4.23) |
| | **22.82 (1.55)** | 44.38 (1.41) | 232.94 (9.89) | 96.2 (3.99) |
| | 22.96 (1.40) | 44.42 (1.56) | 233.14 (10.16) | **96.54 (3.82)** |
| | 23 (1.48) | 44.44 (1.51) | 233.24 (9.83) | 96.54 (3.92) |
| | 23.04 (1.46) | 44.58 (1.61) | 233.28 (10.12) | 96.62 (3.95) |
| | 23.06 (1.53) | 44.64 (1.25) | 233.4 (10.10) | 96.62 (4.09) |
| | 23.16 (1.41) | 44.8 (1.25) | 233.46 (10.29) | 96.76 (4.03) |
| | 23.18 (1.49) | 45.04 (1.46) | 233.54 (9.64) | 96.86 (3.96) |
| | 23.38 (1.60) | 45.96 (1.34) | 233.88 (9.74) | 97.04 (4.16) |
| | 23.4 (1.62) | 46.84 (1.67) | 233.9 (10.14) | 97.2 (3.88) |
| | 23.48 (1.53) | 47.64 (1.53) | **235.82 (10.08)** | 97.24 (4.19) |