Benchmarking of quantum processors with random circuits

James R. Wootton
Department of Physics, University of Basel, Klingelbergstrasse 82, CH-4056 Basel, Switzerland

Quantum processors with sizes in the 10-100 qubit range are now increasingly common. However, with increased size comes increased complexity for benchmarking. The effectiveness of a given device may vary greatly between different tasks, and will not always be easy to predict from single and two qubit gate fidelities. For this reason, it is important to assess processor quality for a range of important tasks. In this work we propose and implement tests based on random quantum circuits. These are used to evaluate multiple different superconducting qubit devices, with sizes from 5 to 19 qubits, from multiple hardware manufacturers: IBM Research and Rigetti. The data is analyzed to give a quantitive description of how the devices perform. We also describe how it can be used for a qualititive description accessible to the layperson, by being played as a game.

INTRODUCTION

The state of n bits resides within a space of 2^n bit strings. By charting a suitable course through this space, classical computers can solve virtually any problem.

The state of n qubits is described by a 2^n dimensional Hilbert space. This more general structure allows a new and more subtle ways to move around the space, giving us new and more efficient routes from input to output. The fact that this will allow us to solve certain problems much faster is the primary motivation behind quantum computation.

To determine whether any given quantum processors can live up to this promise, they need to be benchmarked. This could be done using techniques specifically designed for the task, such as randomized benchmarking []. It could also be done by performing test instances of important quantum algorithms [] or error correction []. The results of such tests will depend both on the noise levels of the device and also its size and connectivity. The insights gained will therefore be highly dependent on the details of implementation, with the results from a given instance of a given algorithm not providing an unambigous predictor for the results of others.

To supplement such results, we can seek a task which is more universal in scope. One that can be implemented on devices of any size and connectivity, and which directly tests the most important primitive for quantum computing: the ability to fully explore the multiqubit Hilbert space.

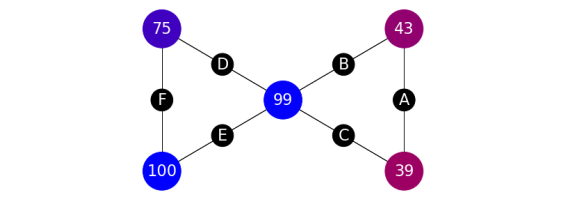
This can be done using random quantum circuits [**?**]. By implementing random progams, the resulting output states are random samples from the Hilbert space of the device. For short depth random circuits, this sampling will be of states with short range entanglement that are close to the product states. But for sufficiently long circuits, which allow for the build-up of entanglement across the device, the states will be sampled uniformly from across the entire Hilbert space. Measuring the qubits will then generate bit strings according to a Porter-Thomas distribution, which provides an observable signature of this quantum chaotic regime [**?**].

The main application of such sampling will be to act as a test of computational power. Entering into the Porter-Thomas regime for a sufficiently large quantum device would allow a demonstration of quantum computers outperforming classical computers: a milestone known as *quantum computational supremacy*. The size of devices needed to acheive this will be relatively large [], and will be well beyond the devices considered in this study. Nevertheless, analysis of random circuits for smaller devices will help benchmark our progress towards this milestone, as well as towards the longer term and more important goal of scalable and fault-tolerant quantum computation.

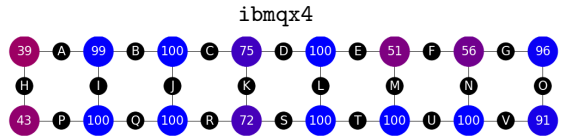
GENERATION OF RANDOM CIRCUITS

For any given device, we will have a set of native gates to work with. These will include arbitrary single qubit rotations, and entangling gates. The latter are typically two qubit controlled operations, such as the controlled-NOT or controlled-Z. In general, it will not be possible to directly implement these between any given pair of qubits on the device. Instead we will have a connectivity graph, in which qubits are nodes that are connected only by an edge when direct coupling is possible. For most physical implementations of qubits, the most straightforward connectivity to realize is a line, for which qubits can couple directly only to their two neighbours. The most powerful and flexible connectivity would be a complete graph, in which each qubit can couple with any other. A good compromize between these extremes would be a planar lattice, such as a square lattice, as required for the implementation of the most prominent error correcting codes []. The connectivity graphs for the real devices used in this study are shown in Fig. 1.

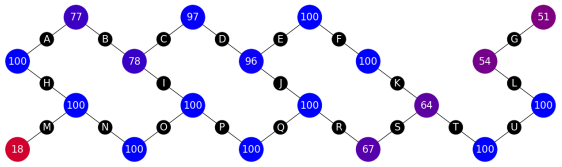
Random circuits are generated by sampling gates from this gate set. The exact form of this sampling will be designed for specific requirements, such as ease of implementation and the acheiving the fastest possible build-up of long range entanglement [**?**]. The better the connectivity of a device, the less restrictions will be provided by



(a) 5 qubit IBM device



(b) 16 qubit IBM device `ibmqx5`



(c) 19 qubit Rigetti device `19Q − Acorn`

FIG. 1: Coupling graphs for devices studied in this work. Coloured circles denote qubits. The lines between them (labelled with letters) denote pairs for which entangling gates can be performed. The numbers within the coloured lines show an example set of results for the circuit, with each representing $\tilde{\theta}_j$ $(\pi/2)$ (as defined in Eq. (5)) expressed as a percentage.

the former consraint, allowing the latter to be done more effectively.

In this work, we introduce a novel method of sampling random circuits. This is done such that: (i) the rate at which entanglement builds up can be make as slow as desired, and (ii) the output possesses easily recognizable structure for states with only simple entangled states. This allows us to gain insights by comparing runs with different rates of entanglement generation, and use the loss of the structure in the output to assess how the entanglement build-up occurs.

To do this, we build up circuits as a series of *rounds*. Each of these is composed of a pair of *slices*, known as the entangling slice and the inverse slice. The former is a randomly generated set of gates that entangle disjoint pairs of qubits, while the latter is an attempt to invert this. The quality of these attempted inversions determine how fast the entanglement builds up. In the extreme the inversions are perfect, the state will always maintain only short-range entanglement. In the extreme the inversions are also randomly generated, they will provide extra random entangling gates to accelarate the build-up of long range entanglement.

Typically, we consider the inversion gates to be chosen based on output data from an implementation of the circuit so far. Specifically, the first run of the circuit uses only the first slice of the first round. The results are then used to define the second slice of the first round. The next run then implements the whole of the first round, followed by the first slice of the second. The results are used to define the second slice of the second round, and so on.

The random generation of gates in each entangling slice is done by first randomly choosing a set of disjoint pairs of qubits. This pairing should be based on the connectivity of the device used: it should be possible to directly implement a controlled gate between the two qubits of each pair. In the case that the device has an odd number of qubits, or if the connectivity requires it, some qubits can be left unpaired.

Next, a random entangling gate is generated for each pair of qubits (j, k). This will take the form

$$\mathtt{cx}(j,k)\, \exp[\, i\, \theta_{jk}\, \sigma_x^j\,]\, \mathtt{cx}(j,k) = \exp[\, i\, \theta_{jk}\, \sigma_x^j \sigma_x^k\,] \qquad (1)$$

Here $\mathtt{cx}(j,k)$ denotes a controlled-NOT with j as control and k as target. The angles θ_{jk} are chosen randomly from the range $\pi/20 \leq \theta_{jk} \leq \pi/2$. The effect of these gates on an initial state with $|\,0\rangle$ for all qubits will be to create entangled pairs of the form

$$\cos(\theta_{jk})\,|\,00\rangle + i \sin(\theta_{jk})\,|\,11\rangle\,. \qquad (2)$$

For such pairs, note that Z basis measurement of the two qubits will always yield the same result. The probability that this result is $|\,1\rangle$ is

$$p_j = p_k = \sin^2(\theta_{jk}). \qquad (3)$$

Since the θ_{jk} are restricted to the range $\pi/20 \leq \theta_{jk} \leq \pi/2$, the values of p_j will yield values of p between 0.024 and 1. The lower bound was chosen to ensure a degree of distinction between qubits involved in pairs (and therefore an entangling gate) and those left unpaired (and therefore with no gate applied).

Given this structure in the output, it should be possible to deduce the random gates applied using only the values of p_j for each qubit. These can be used to deduce the pairing using the fact that $p_j = p_k$ for each pair (j, k). The value of θ_{jk} can then be deduced by inverting Eq. 3. Since this completely specifies the gates of the entangling slice, it can be used to construct the corresponding inverse slice.

It is important to note the deduced inverse will not be a true inverse in general. Reasons for this include:

- Noise in the implementation, such as imperfect gates and decoherence, will perturb the measured p_j from their ideal values;
- The finite number of samples used to estimate the p_j will lead to statistical noise;

- The use of a faulty method to construct the inverses;
- Failures in the inverses of previous rounds will result in the entangling slice not being applied to the all $|0\rangle$ state, and so Eq. **??** applies only approximately.

In preventing the first slice of each round from being fully inverted, these effects allow randomness to build up in the circuits. By choosing how strong these effects are, we can tune the rate at which entanglement builds up throughout the circuit.

Note that each round, as defined thus far, is completely diagonal in the σ_x basis of all qubits. Using oly such gates will not allow us to fully explore the Hilbert space of the device. The finishing touch for each round will therefore be to conjugate completed rounds with random single qubit gates. Each of these is randomly chosen to be either an x or y axis rotation, and with a randomly chosen angle $0 \leq \phi \leq \pi$.

FIGURES OF MERIT

With the results from running the circuit for each round we can assess the build-up of entanglement in a device. This will primarily be done by looking at how well the output can be used to deduce the inverse of the most recent slice of randomly chosen entangling gates. Highly successful construction of the inverse implies that long range entanglement is negligible, and that the state immeditaley prior to the most recent slice was close to the all $|0\rangle$ state. The relation of Eq. **??**, and all conclusions derived from it, will then hold to good approximation.

On the other hand, highly unsuccessful construction of the inverse implies that final output is dominated by other effects. In the best case, this will be long range entanglement built up by the random circuit. In the worst case, it will be noise. By comparison of random circuits for which entanglement is generated at different rates, we can attempt to distinguish these to possibilities.

Note that we will use $\tilde{p_j}$ to denote the measured probability of qubit j to output the result $|1\rangle$. This is distinct from p_j in general, because of the effects of the imperfections listed in the previous section.

Fuzz

The first way we will quantify an ouput is to compare the calculated values of $\tilde{p_j}$ and $\tilde{p_k}$ for each pair in the most recent slice. If Eq. **??** holds, we will have $\tilde{p_j} = \tilde{p_k}$ in each case. However, as Eq. **??** becomes an increasingly worse approximation, these numbers will begin to drift away from each other. We refer to this as *fuzz*, and quantify it as follows over the whole device

$$\texttt{fuzz} = \sum_{(j,k)} |\tilde{p_j} - \tilde{p_k}|/n. \tag{4}$$

Here n is the number of pairs of qubits on the device (and so half the total number of qubits when this is even and the connectivity allows).

Note that, for the first round, the fuzz will be at or close to zero. It will then begin to rise as Eq. **??** becomes more approximate. At the other extreme, after an arbitrarily large number of rounds, all $\tilde{p_j}$ will converge at close to 1/2. This will ideally be due to the random circuit causing the final state to be a typical sample drawn uniformly from the multiqubit Hilbert space. However, it could also be due to the build up of noise. In either case, the fact that all $\tilde{p_j}$ have converged to the same value will also cause a low value of the fuzz.

Given this behaviour, a graph of fuzz against round number will neccesarily feature a peak. This will be the most noticable feature in our results. It essentially marks the start of the inevitable march towards a completely random output without the structure required for inverses to be successfully deduced.

We will look at the build-up of fuzz for each device in two different cases: (i) inverses constructed when the assumed pairing of the qubits is completely correct, and the deduced angles are correct up to effects caused by statistical noise and the build-up of entanglement, and (ii) inverses constructed when the assumed pairing is chosen completely randomly and without reference to the results. When case (i) is run on a real device the $\tilde{p_j}$ cannot be used to deduce the θ_{jk} since additional noise effects would also be present. The effects of statistical noise alone is therefore emulated by taking the correct values and adding $0.1/\sqrt{\texttt{shots}}$, where `shots` is the number of samples used for $\tilde{p_j}$. For simulated instances of case (i), and for case (ii), the assumed θ_{jk} are calulated from Eq. **??** using $(\tilde{p_j} + \tilde{p_k})/2$.

In the absence of noise, the inverses for case (i) are perfect up to statistical noise. This can be suppressed arbitrarily by increasing the number of samples used to calculate the $\tilde{p_j}$. For case (ii), however, the fuzz will rise sharply and peak early. This is because the second slice of each round is essentially as much a source of random gates as the first, and has little effect as an inverse.

By looking at the fuzz for these two cases, we can assess how well a device a device might be able to uniformly sample states from its multiqubit Hilbert space. Let us consider this to be done by using case (ii) (the inverse slices with randomly chosen pairs), since this would provide the fastest buil-up of entanglement.

The creation of the desired states requires the fuzz to first peak, and then subsequently vanish. However, this same behaviour will also be seen for devices dominated by noise. To determine which of these two possibilities

occurred, we can run the process again over the same number of rounds, but instead use case (i) (the inverse slices with correctly chosen pairs). If the noise is dominant, the fuzz will again be seen to peak and vanish. If noise is neglible, however, and a large value of `shots` is used, the increase in fuzz will be only very slight.

Ideally, we would like to see the fuzz remain at a low and pre-peak value for case (i) for as many rounds as it takes for the fuzz of case (ii) to first peak and then subsequently vanish. This is because the fuzz of case (i) is primarily caused by noise, and so low values imply that noise remains at low levels. This provides strong evidence that the vanishing fuzz for case (ii) is primarily due to the build up of entanglement.

For devices unable to acheive this condition, we can also define a weaker goal. This is simply that the fuzz should peak for case (i) at a higher round than for case (ii). We will specifically require this for runs on the real device in the former case (when fuzz is likely dominated by noise), and for a simulated run in the latter case (noiseless, and so fuzz built-up is dominated by the random inverse slices). If this condition is not satisfied, it means that the strength of noise in the former case is stronger than the incompetence of the inverses in the latter case. By demonstrating that the effects of noise in a device are not at such a high level, we could say that it has acheived *quantum competence*. This milestone must obviously be passed significant before that of quantum supremacy.

Success rate for pairing

We will now consider how well the pairing can be deduced for a given output. We will do this using minimum weight perfect matching (MWPM) [] on the connectivity graph of the device. For each pair of qubits we assign a weight that depends on the measured values of $\tilde{p_j}$,

$$W_{j,k} = |\tilde{\theta_j} - \tilde{\theta_k}|, \ \tilde{\theta_j} = \sin^{-1}(\sqrt{\tilde{p_j}}). \tag{5}$$

The minimum weight matching will find the pairing that minimizes this weight, and therefore minimizes the differences between the $\tilde{p_j}$ values. Since the $\tilde{p_j}$ values for the two qubits within each pair should be equal, peforming this minimization should provide a near optimal means of finding the pairs. The percentage of pairs correctly found my this method will be used as a further way of analysing the progess of our random circuits.

Difference with ideal values

The main result taken from the output is the measured probabilities $\tilde{p_j}$. It therefore makes sense to compare these directly to their ideal values p_j. Specifically, we will calculate the difference between the corresponding $\tilde{\theta_j}$ and the actual angle θ_{jk} used for the pair that qubit j is a part of. This will also be averaged over the entire device to give a measure of how well the qubits correspond to the values they would take if only the most recent entangling slice was implemented, and implemented perfectly.

Error mitigation

Thus far we have used only one of the properties of the states described in Eq. 2: the fact that $p_j = p_k$. However, it is further true that the results for paired qubits j and k should be perfectly correlated. This provides a further means by which pairs can be identified from the data. Specifically, the mutual information $I(j;k)$ for measurement outcomes will be non-zero if and only if the qubits are paired. For a given qubit j, the most likely qubit k for it to be paired with is therefore that with the highest value of $I(j;k)$. Let us refer to this as qubit $c(j)$.

This information could then be used mitigate for effects that cause violations of Eq. **??**. Specifcially, instead of using the measured values of the $\tilde{p_j}$, we instead use

$$\bar{p_j} = \frac{\tilde{p_j} + \tilde{p_k}}{2} \tag{6}$$

For cases in which two qubits are each most correlated with the other (i.e. $c(j) = k$ and $c(k) = j$), the resulting values of $\bar{p_j}$ and $\bar{p_k}$ will be equal. Assuming that the mutual informations can be used to correctly deduce pairings in most cases, this will result in significant improvements to results.

Quantum Awesomeness

Determining the most likely pairing of the qubits given the $\tilde{p_j}$ (or error mitigated $\bar{p_j}$) is a puzzle to be solved. Indeed, it is a puzzle that can be played even without knowledge of the underlying quantum programming. Our scheme then becomes an accessible puzzle game.

If played directly on a device (real or simulated) the pairing supplied by a player will be used to construct the inverse slice for each round. This means that any mistakes made will have an effect on all subseqeuent rounds. This, as well as other effects which cause the build-up of long range entanglemnt or noise, will cause the puzzles to increase in difficulty for each successive round. The players aim will then be simply to keep the game playable for as long as possible.

The game can also be played using saved data, such as that from a run in which the pairs of the inverse slice are always chosen correctly. In this case, the main purpose of the game is to serve as a qualitative way of benchmarking devices. Greater size and more complex connectivity will allow more challenging puzzles, whereas greater noise

will cause an infuriating degree of difficulty. The quality of a given device will therefore correlate well to how enjoyable the game is when played on it. This allows a high-level means of comparing devices that is accessible by the interested lay person.

Examples of what is game would look like for the devices considered in this work are shown in Fig. 1. In these, the number shown on each qubit is the corresponding $\tilde{\theta_j}$ $(\pi/2)$ (as defined in Eq. (5)) expressed as a percentage. These numbers also determine the colour used for each qubit, ranging from blue for 0% to red for 100%. The aim is therefore to identify the correct pairs (which are labelled by letters) by matching qubits with similar numbers and colours.

This game, which is called *Quantum Awesomeness*, can be played with the data presented in this paper at [**?**].

RESULTS

Results were taken for a selection of real and example devices, with both real and simulated data. For runs on real devices, data is taken only for the case of correct pairings with a large number of shots (`shots` ~ 10000). This is then compared to simulated data for the case with random pairings, and for that of correct pairings but far fewer shots (`shots` $= 100$). Ideally, the results should show a build-up of entanglement that is slower than for both the simulated instances. It should especially be much less than for the simulated case of random pairing.

To provide a good understanding of how the figures of merit should behave, we first consider simulated results from a set of example devices.

Example devices

The quantitative benchmarks of the previous section were applied to a set of example devices of different sizes and connectivities. The connectivity graphs considered were a line (with 5, 11, 15 and 19 qubits), a ladder (4, 10, 16 and 20 qubits), a square lattice (4, 9 and 16 qubits) and a fully connected graph (5, 11, 16 and 19 qubits). The qubit numbers where chosen to span the same range as the real devices we will consider, given the constraints of the connectivity (square numbers only for the square lattice, etc).

Results for the fuzz are shown in Fig. **??**. For each connectivity, the fuzz for the smallest case was found to behave very differently than the others. This is due to the fact that the fuzz will converge to a value that decays exponentially with qubit number as the state fully explores the Hilbert space. So though it can be said to vanish for large devices, it will not for small ones. The peak for the fuzz is therefore less visible for such small sizes.

For the larger devices, the graphs show little variation for different sizes over the range considered. This is despite the fact that the build-up of long range entanglement will scale with system size [**?**]. This shows that our figures of merit benchmark the entanglement moving beyond the simple pairing provided by the entangling slices, rather than it becoming truly long range. This means that the vanishing of the fuzz, for example, is a neccessary condition for long range entanglement rather than a witness that it has occurred.

The peak and subsequent decay of the fuzz is found to depend strongly on connectivity. These processes are slowest for the least connected devices (the lines) and fastest for the most connected devcies (fully connected). The ladders and square lattices show similar behaviour, which lies between the two extremes.

Results for the fraction of correct pairings for MWPM are shown in Fig. **??**. In each case, this fraction is found to decrease sharply for the first few rounds, before converging at a value which reflects the fraction of pairs that would be correct for a random guess. The round at which this occurs appears to correspond well to that at which the fuzz peaks.

Similar behaviour is found for the difference between the $\tilde{\theta_j}$ and their ideal values, as shown in Fig. **??**. It first rises sharply before showing signs of convergence. The peak of the fuzz is again found to be a good rule of thumb for the point at which this occurs.

5 and 16 qubit IBM devices

Results for the the 5 qubit IBM device `ibmqx4` are shown in Fig. **??**. The small size of the device, and associated finite size effects, make it difficult to identify features such as the fuzz peak. One distinct feature, however, is that the fuzz for error mitigated results corresponds well to the simulated results for correctly chosen inverse slices up until round 9. Similar agreement is found for the success rate of MWPM up until round 5. Such agreement is not seen for the average difference between the $\tilde{\theta_j}$ and their ideal values. However, this is found to be smaller for the real device than for the randomly chosen inverses in most cases. 0.4

Results for the the 16 qubit IBM device `ibmqx5` are shown in Fig. **??**. We find that the fuzz peaks at round 2, which is extended to round 4 when the error mitigation is used. Both occur earlier than the peak for randomly chosen pairs for the inverse slices, which occurs at round 5.

The decay of the success rate for MWPM occurs over a similar number of rounds. The main decay continues until around round 4. At this point it begins to converge at around 0.4, which is the success rate for random

guessing. The success rates for the real device are found to be very similar to that of the simulation for randomly chosen pairs for the inverse slices

The error mitigated data decays much more slowly, maintaining success rates of around 0.7 as far as round 10. The success rates are much highly than those for the simulation of randomly chosen pairs for the inverse slices. However, they are still much less than the simulation of correctly chosen pairs with low shots, which still remains above 0.9 at round 10.

The average difference between the $\tilde{\theta}_j$ and their ideal values is found to be higher than that for randomly chosen pairs until round 6. Little difference is seen between non-mitgated and mitigated data, because our method of mitigation does not aim to correct for these values.

19 qubit Rigetti device

Results for the the 19 qubit Rigetti device `19Q − Acorn` are shown in Fig. **??**. Here we find that the fuzz starts of at a peak at round 1, and decays thereafter. Error mitigation greatly reduces the values of the fuzz, though it also makes it difficult to distinguish the point at which the peak occurs. Nevertheless, it does seem to be delayed until at least round 3 by the mitigation. In either case, it is earlier than the peak for randomly chosen pairs for the inverse slices, which occurs at around round 8.

The success rate for MWPM remains at around 0.6, which is the success rate for random guessing, for all rounds studied. For mitigated data, however, the success rate starts as high as round 0.9 and decays rapidly over the first four rounds. These success rates are found to be very similar to those of the simulation for randomly chosen pairs for the inverse slices.

The average difference between the $\tilde{\theta}_j$ and their ideal values is noticably higher than that for randomly chosen pairs up to round 10, where we cease to take data. Again, little difference is seen between non-mitgated and mitigated data, because our method of mitigation does not aim to correct for these values.

CONCLUSIONS

We compared all quantum processors currently publicly available using benchmarks based on random circuits. The process can essentially be viewed as a game played on the devices, with the game designed such that it remains playable over many round only when the player is skilled.

Runs implemented on real devices were for the case of a perfect player. These were compared with simulated (and therefore noiseless) runs on the same device for both a similarly perfect player, and one whose moves are entirely random.

It was found that the game when run on a real device corresponds well to the simulated runs with a random player. This demonstrates the strength of noise present in these devices: The glitchiness they cause in the game causes the same difficulty level as a completely unskilled player.

Based on this, we define the notion of *quantum competence*. To acheive this, a device should perform better for a perfect player on a real device than for an incompetent player on a perfect device. We found evidence that this is indeed acheived for the IBM devices, but only when post-processing is run on the output to mitigate for errors.

The conditions required for quantum computational supremacy seem to still be beyond current devices. The milestone of quantum competence is therefore a more realistic goal for experimental efforts in the near-term. Devices should be developed to show ever stronger demonstrations of this goal, and maintain this quality while the size of the devices is increased.

ACKNOWLEDGEMENTS

This work was supported by the Swiss National Science Foundation and the NCCR QSIT.

Results for this work were generated using hardware from IBM Q and Rigetti, and software from IBM Q (QISKit), Rigetti (Forest) and Project Q. The views expressed are those of the author and do not reflect the official policy or position of any of these entities.

Figures for results

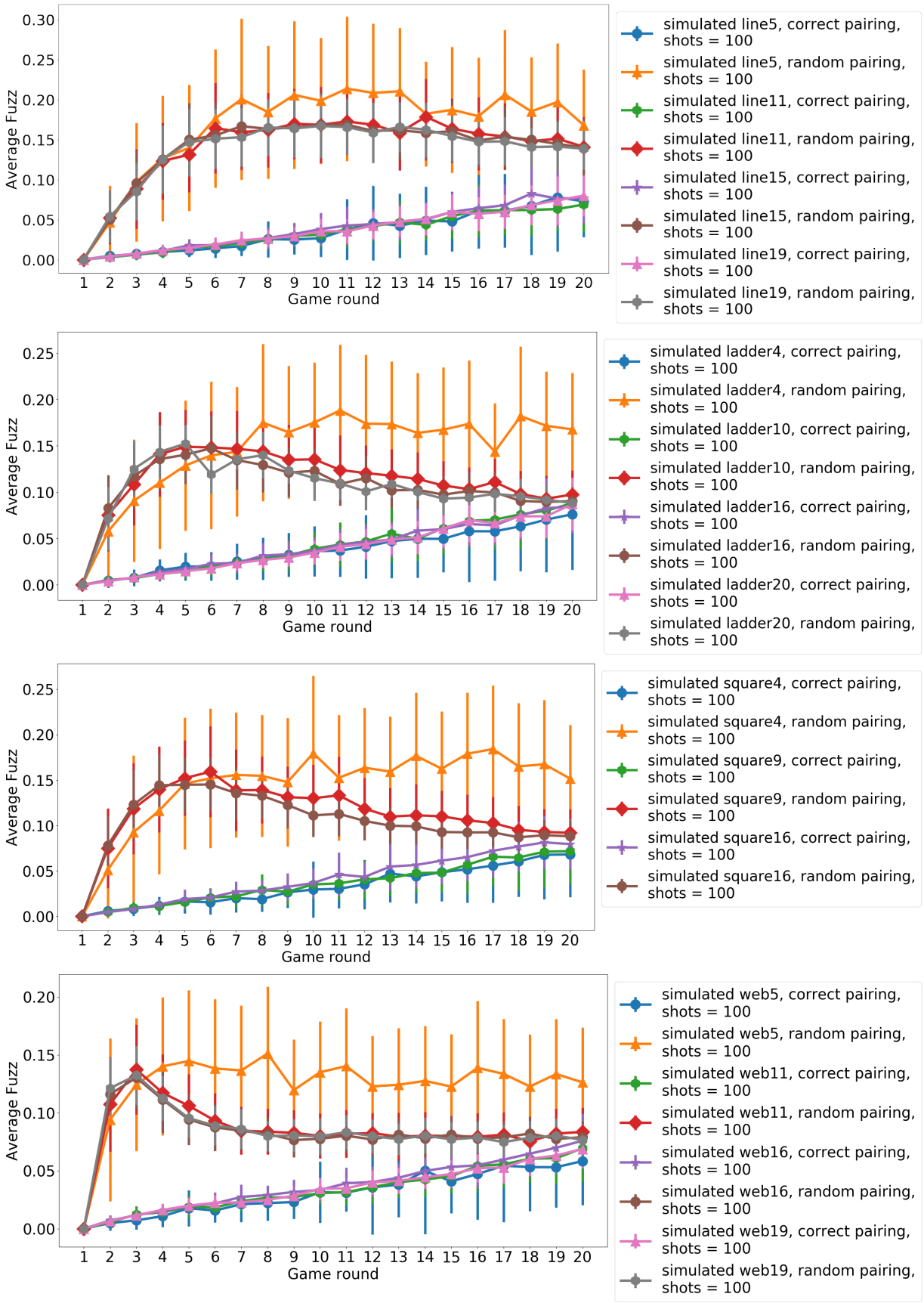


FIG. 2: The average fuzz for all example devices. Each point is the average of 100 samples, with error bars given by the standard deviation. These results are discussed in section **??**.

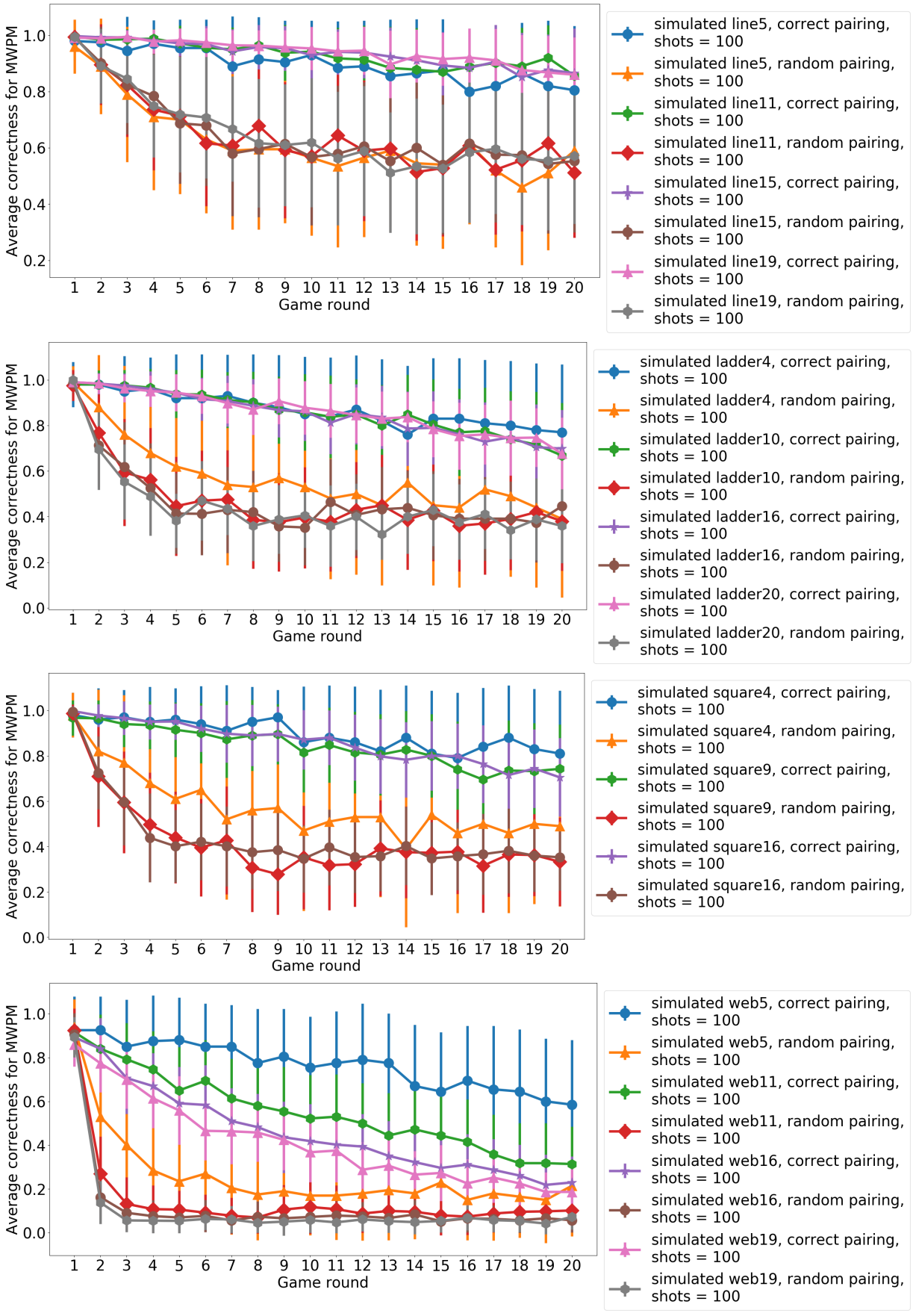


FIG. 3: The average correctness of pairing via minimum weight perfect matching for all example devices. Each point is the average of 100 samples, with error bars given by the standard deviation. These results are discussed in section **??**.

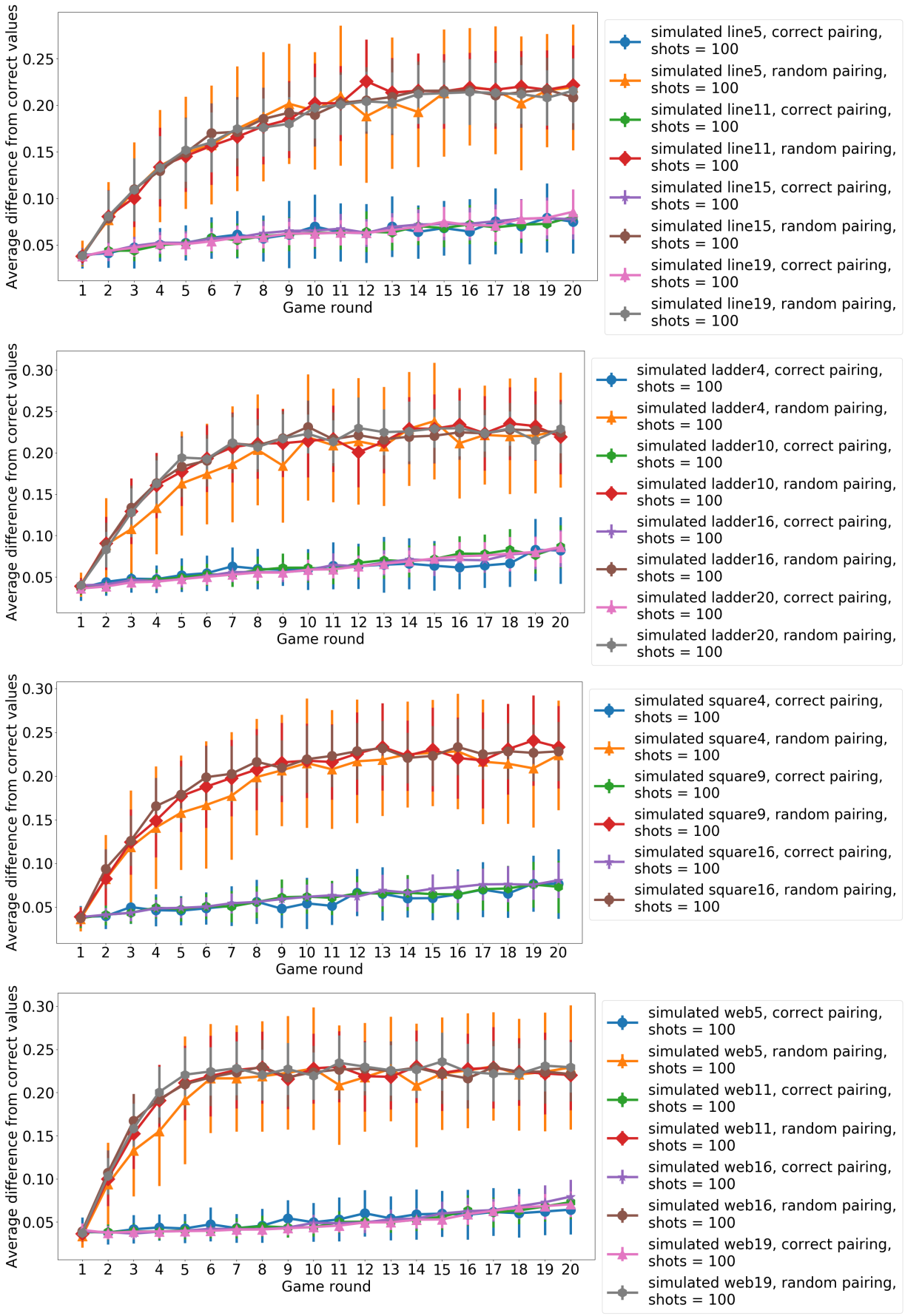


FIG. 4: The average difference between inferred and correct values for the θ_{jk} for all example devices. Each point is the average of 100 samples, with error bars given by the standard deviation. These results are discussed in section ??.

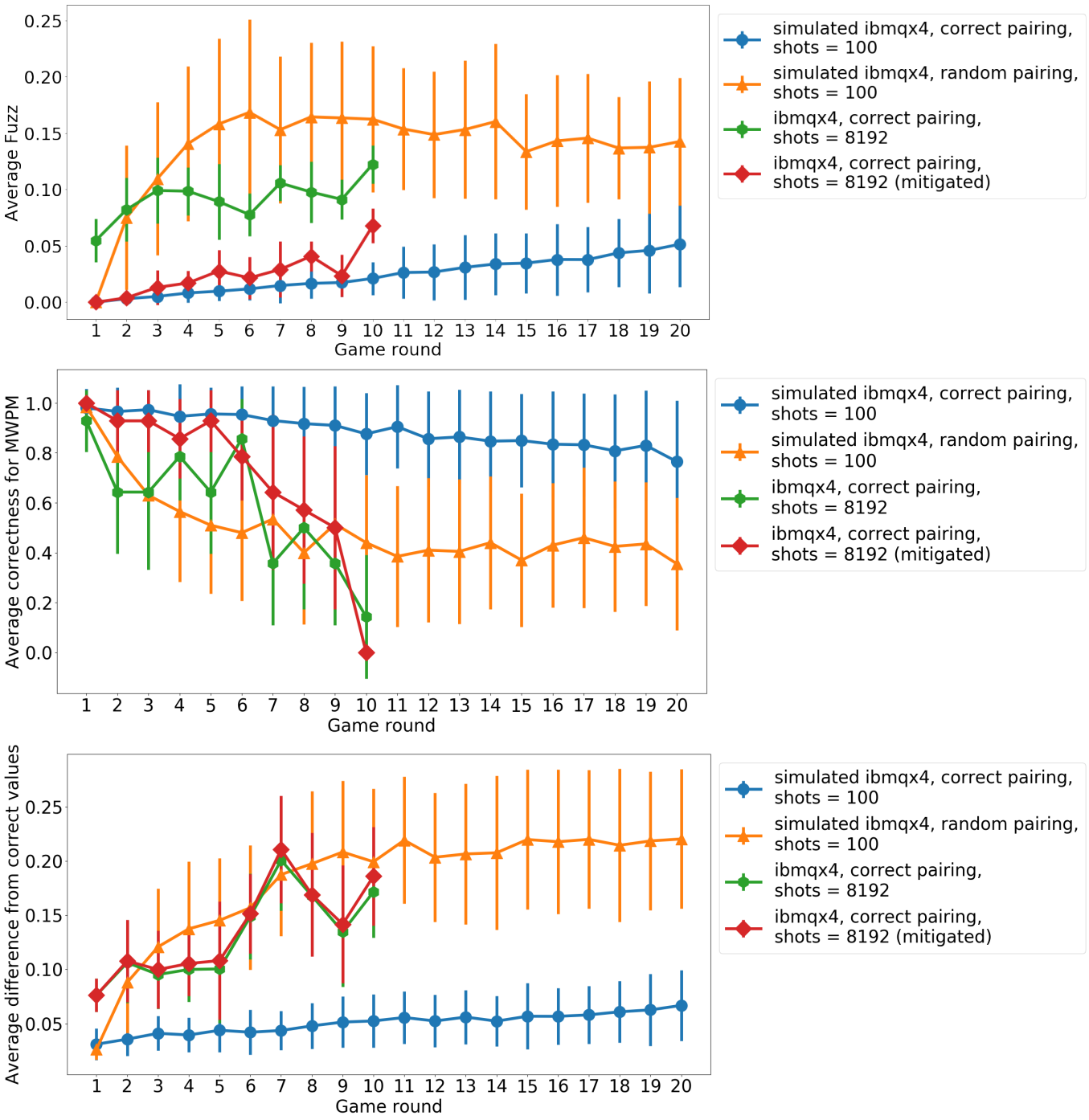


FIG. 5: Results for the IBM device `ibmqx4`. Each point is the average of 100 samples for simulated data and around 10 for the real device (more will be added soon). Error bars given by the standard deviation. These results are discussed in section **??**.

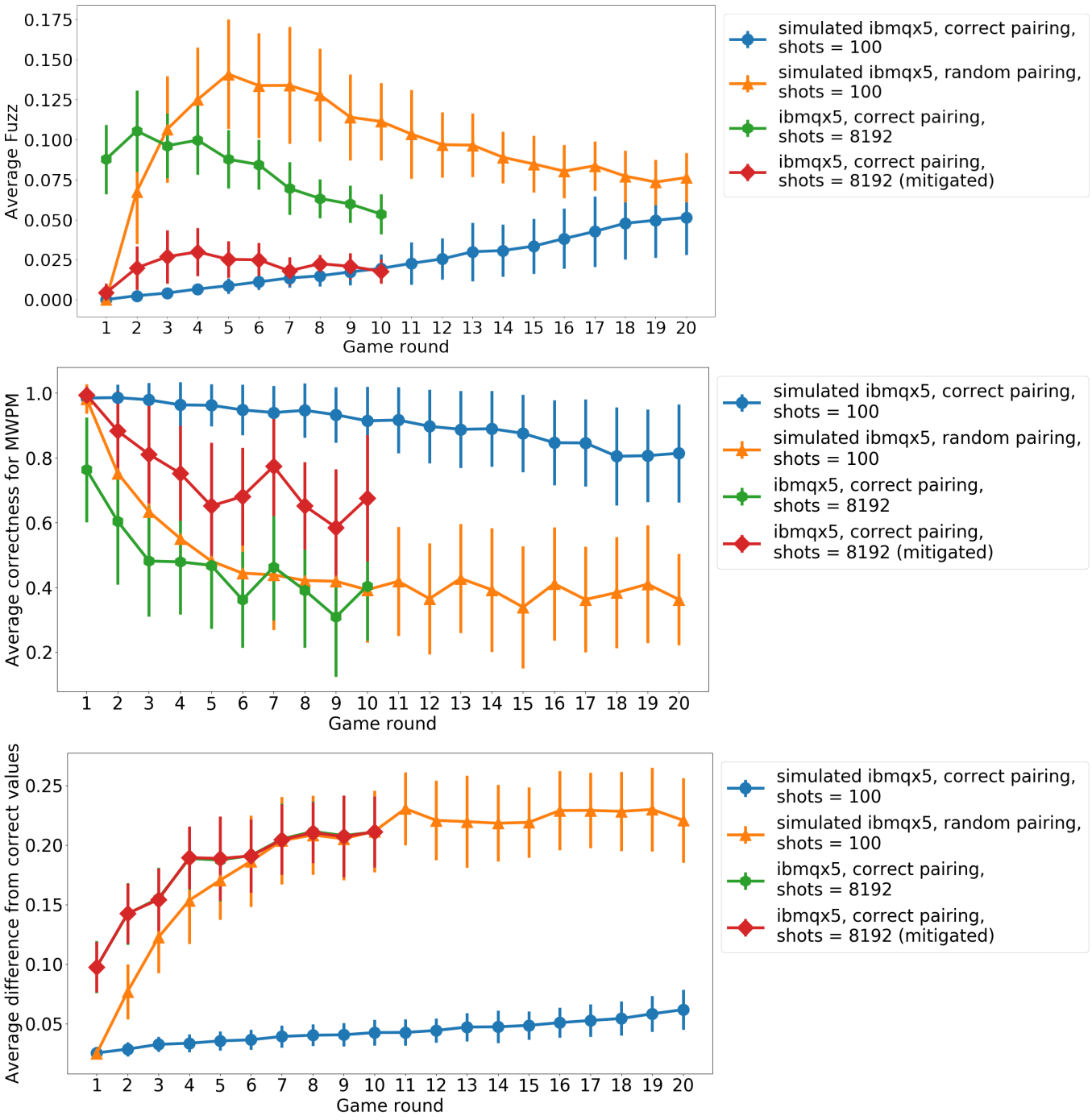


FIG. 6: Results for the IBM device `ibmqx5`. Each point is the average of 100 samples for simulated data and around 50 samples for the real device. Error bars given by the standard deviation. These results are discussed in section **??**.

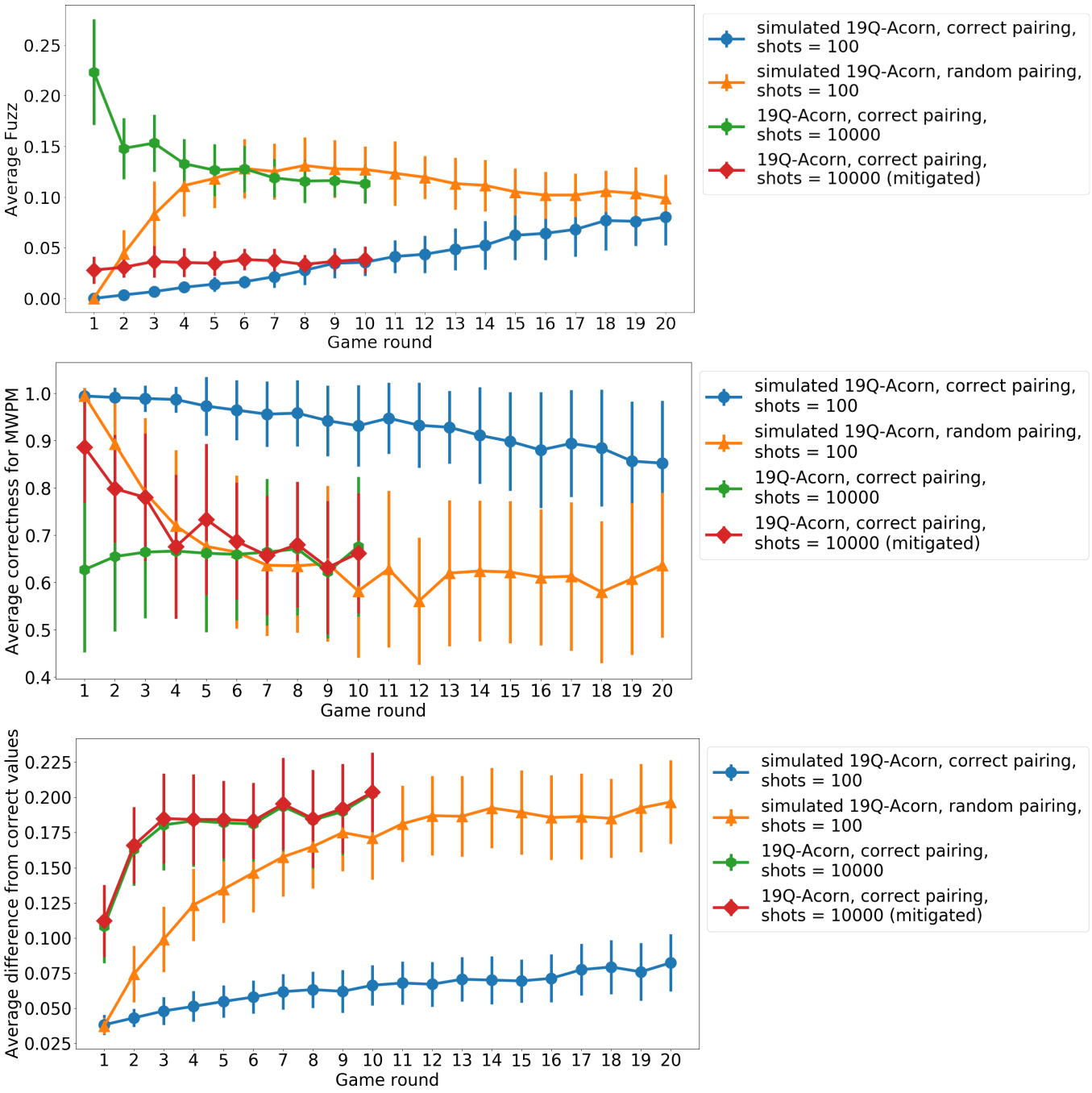


FIG. 7: Results for the Rigetti device `19Q − Acorn`. Each point is the average of 100 samples for simulated data and around 50 samples for the real device. Error bars given by the standard deviation. These results are discussed in section **??**.