JEnt (MemOnly)

and JEnt Analysis Approaches

Joshua E. Hill

KeyPair CONSULTING

Version: 20221018

Relevant [90B] and [IG] D.J, D.K shall statements

#5: Although the noise source is not required to produce unbiased and independent outputs, **it shall exhibit random behavior; i.e., the output shall not be definable by any known algorithmic rule.**

#23: The documentation shall **justify why the entropy source can be relied upon to produce bits with entropy**.

#36: Documentation shall provide an explicit statement of the expected entropy provided by the noise source outputs and **provide a technical argument for why the noise source can support that entropy rate**.

#37: The **noise source state shall be protected from adversarial knowledge or influence** to the greatest extent possible. The methods used for this shall be documented, including a description of the (conceptual) security boundary's role in protecting the noise source from adversarial observation or influence.

Relevant [90B] and [IG] D.J, D.K shall statements

#35: (*OPTIONAL)* Documentation shall include **why it is believed that the entropy rate does not change significantly during normal operation**.

#116-117: The technical argument supporting the expected H_submitter value shall be based on the vendor's description of the source of unpredictability… **Statistical testing may be used to establish parameters referenced within this argument, but the H_submitter value shall not be the result of some general statistical testing process that does not account for the design of the noise source.**

Relevant [90B] and [IG] D.J, D.K shall statements

“Account for the design of the noise source”?

- To use a “**general statistical testing process**”, we need to explain why the statistical test produces a meaningful bound.
- This can be hard in practice…

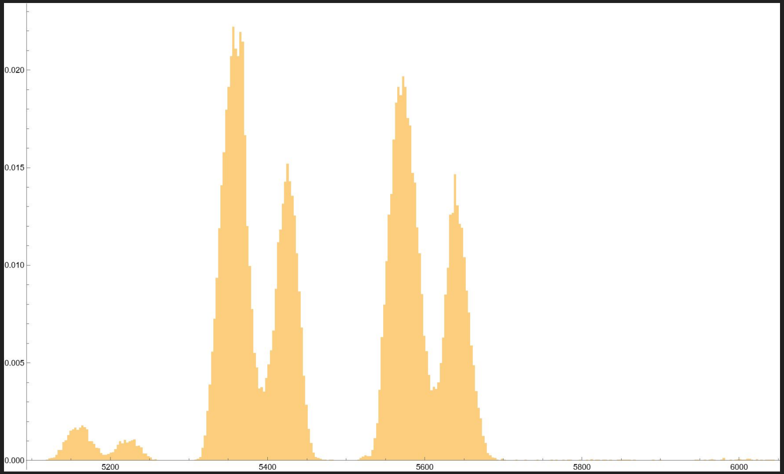
What “non-physical” phenomenon is “noisy”?

The JEnt design paper [Müller 2022] describes various possible causes for unpredictable timing:

- Architectural **features that can impact execution time**
  - **CPU instruction pipeline fill levels** have an impact on the execution time of one instruction.
  - **CPU frequency scaling** can alter instruction processing speed, and may depend on workload.
  - **CPU power management** can disable CPU features that impact execution time.
  - **CPU frequency scaling** depending on the workload.
  - **Translation Lookaside Buffer (TLB)**
  - **Scheduling and load balancing**
  - **Interrupt handling**

What “non-physical” phenomenon is “noisy”?

This is a complex set of effects that varies widely across architectures



A general approach based on these features is difficult to justify

More Noise Means… More… Better?

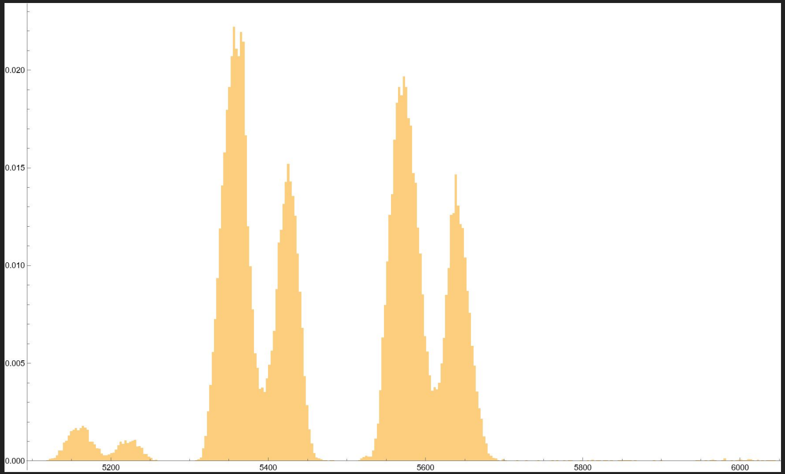
- The data in this source can be difficult to directly test.
- A lot going on yields large counter values.
  - Fast counters yield larger counter values and more variation.
  - More variation suggests there could be more entropy…

But…

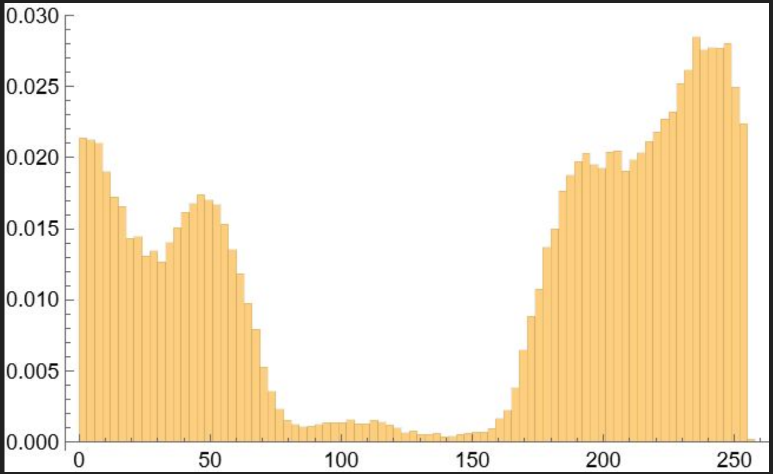
  - Large variation means we can't directly assess the data.
  - Translation just further obscures what is going on…

More Noise Means… More… Better?

If we truncate to the lowest byte, we can run the SP 800-90B estimators…



Becomes



Getting to Know You SP 800-90B Estimator

- These estimators are conceptually simple.
- Many estimators operate under an IID assumption.
- Many estimators essentially extract a single extracted parameter.
- All of the estimators work better when supplied a fixed distribution.

Conclusion: Not magic boxes that output truth.

Problem Statement

- We need a simple source of variation that we think is non-deterministic.
- We would like the resulting data to reflect only the chosen source.
- We would like to be able to minimize or avoid translation or mapping.
- We want to be able to argue that the statistical testing is meaningful.

Proposed JEnt Changes

What “non-physical” phenomenon is “noisy”?

- Architectural **features that can impact memory access timing**

Different types of memory (L1, L2, L3 caches, main memory) are operated with different frequencies which require the introduction of wait states to synchronize the CPU accessing these memory components.

  - **CPU and memory bus clocking** differences
  - CPU has to enter **wait states for the synchronization of any memory access** where the time delay added for the wait states adds to time variances.
  - **Instruction and data caches** with their varying information.
  - **CPU topology and caches used jointly by multiple CPUs.**

A general approach based on memory only allows for a justifiable technical argument

Focus On What You Want

- JEnt raw data currently is influenced by both memory access timing variation and (large) conditioning.
- The large conditioning operation exposes the system to substantial timing variations.
- Memory timing also varies, but this variation is at a more modest scale.
- Prior investigation has identified variations in memory timing as a source of non-determinism.
  - (In some architectures)
- We change the primary noise source to reflect only the memory update timing.
- Other timing variation is still used as an additional noise source.

Take Only What You Want

- Even simple sources have distinct behaviors (thus timing sub-distributions).
- These behaviors reflect how successful the caching system has been.
- We are interested timing updates that result in actual RAM I/O.
- Because the source is simple, we can classify the sub-distribution by timing alone.
- We filter the output of the primary noise source so that it contains only the identified sub-distribution.
- Other data is still used as supplemental data in the conditioner.

BUT WAIT, THERE'S MORE!

In summary:

- The primary noise source now reflects only memory timing (a single sort of event that we think may be non-deterministic… on some architectures).
- We filter the results so that we only output data from a single sub-distribution.
  - We can configure the library so that the desired sub-distribution occurs suitably often.
- Data from the primary noise source now requires less (or no) translation.

Our estimators now have a fighting chance…

(In related news, health testing is similarly more powerful.)

JEnt Codebase Flavors

We have a GitHub branch with this functionality:

[JEnt (MemOnly)]: MemOnly branch with an associated pull request

Assessment Strategies

Assessment Strategy #1
The Single Sub-Distribution Empirical Analysis Approach

- Choose parameters so that RAM I/O is likely.
- Select the desired sub-distribution.
- Test raw data output using the SP 800-90B estimators.

(The “Do the needful” assessment approach.)

Assessment Strategy #2
The Essentially IID Analysis Approach

This starts the same way:

- Choose parameters so that RAM I/O is likely.
- Select the desired sub-distribution.

At this point, engage the statistical fanciness…

Assessment Strategy #2
The Essentially IID Analysis Approach

- Non-IID sources have statistical memory. Internal state that induces relationships between the current output and some number of past outputs.
- The statistical memory "depth" is the number of symbols for which that state induces a significant interrelationship.
- If the memory depth is finite, we can decimate (throw away) enough data so that the remaining data acts like IID data.
  - "Thrown away" data can still be integrated into the conditioner as "supplemental data" and not credited as containing entropy.

How do we know when we've thrown away enough data?

Assessment Strategy #2
The Essentially IID Analysis Approach

Essentially, run the SP 800-90B (Section 5) IID tests… a lot…

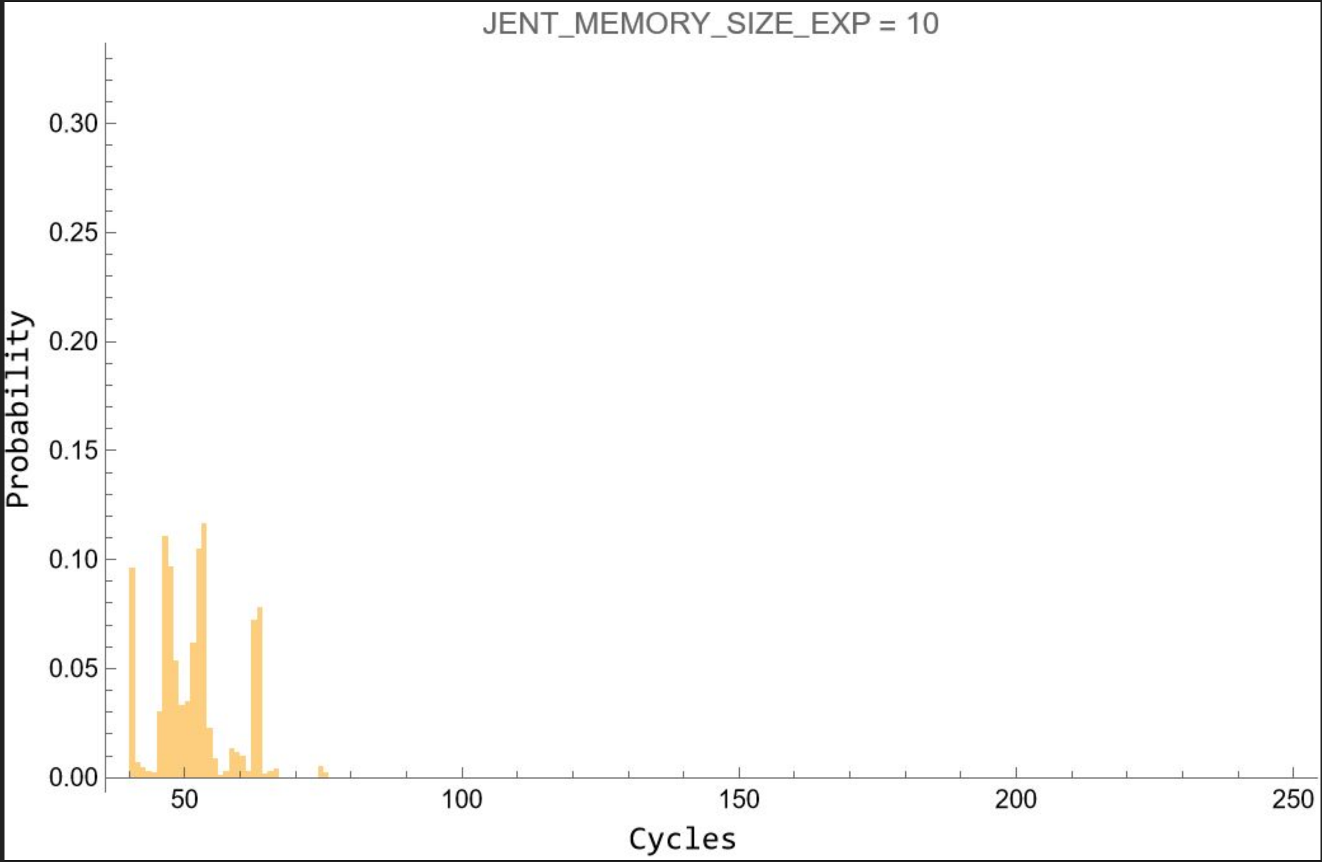
- Take many samples of data.
- Run each of the 22 tests on each of the data samples.
- Check to see if each of the 22 IID tests is passing “sufficiently often”.

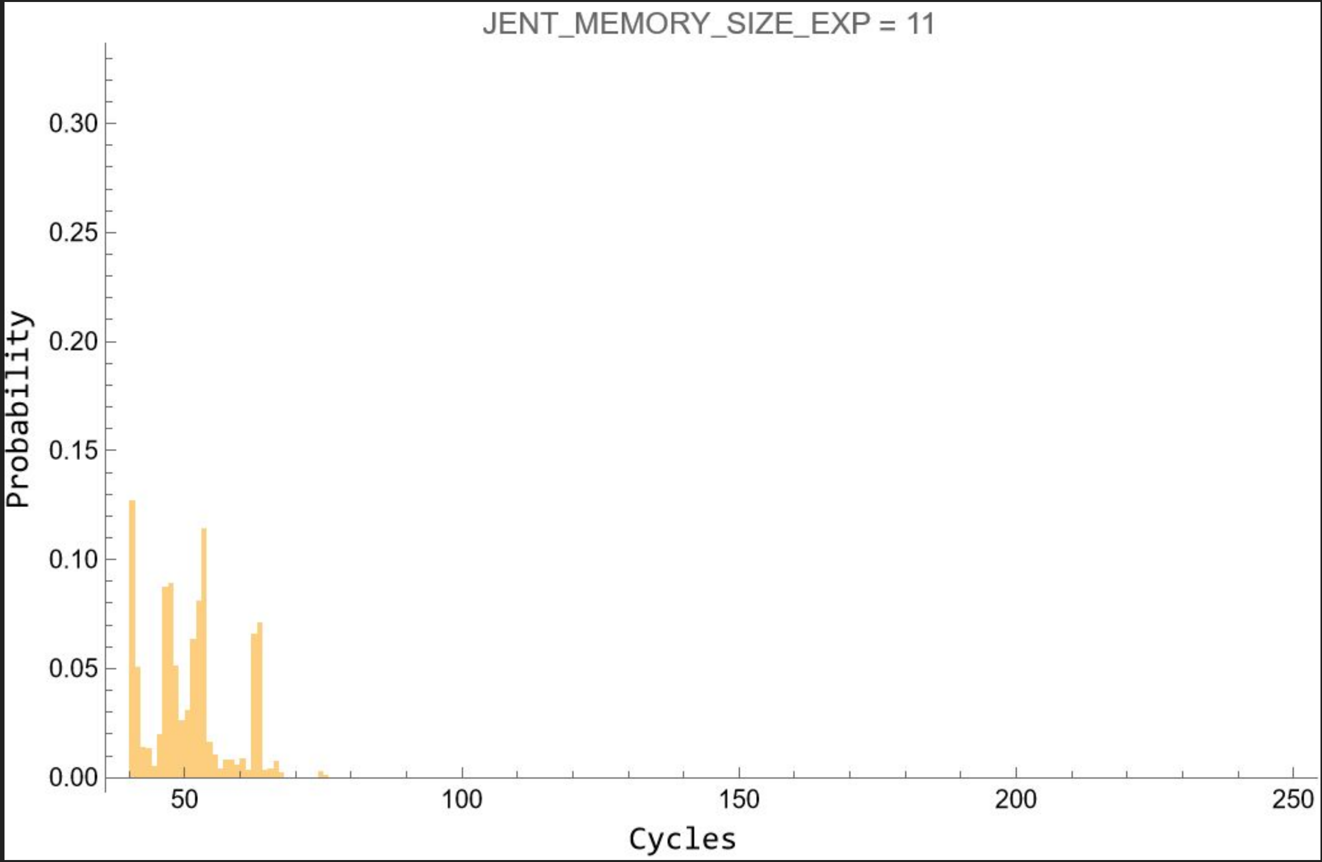
Do this for each decimation level until it works.

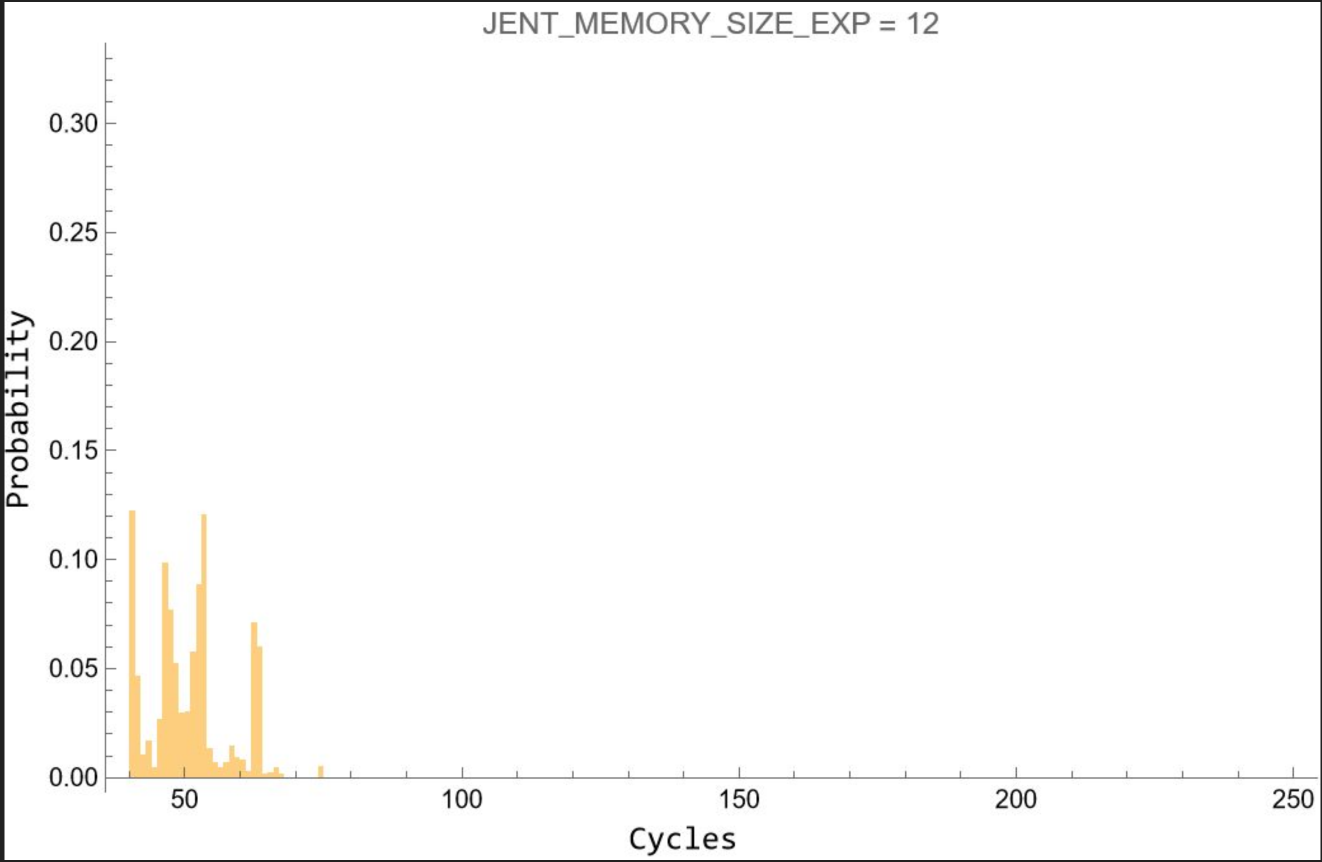
Assessment Strategy #2
The Essentially IID Analysis Approach

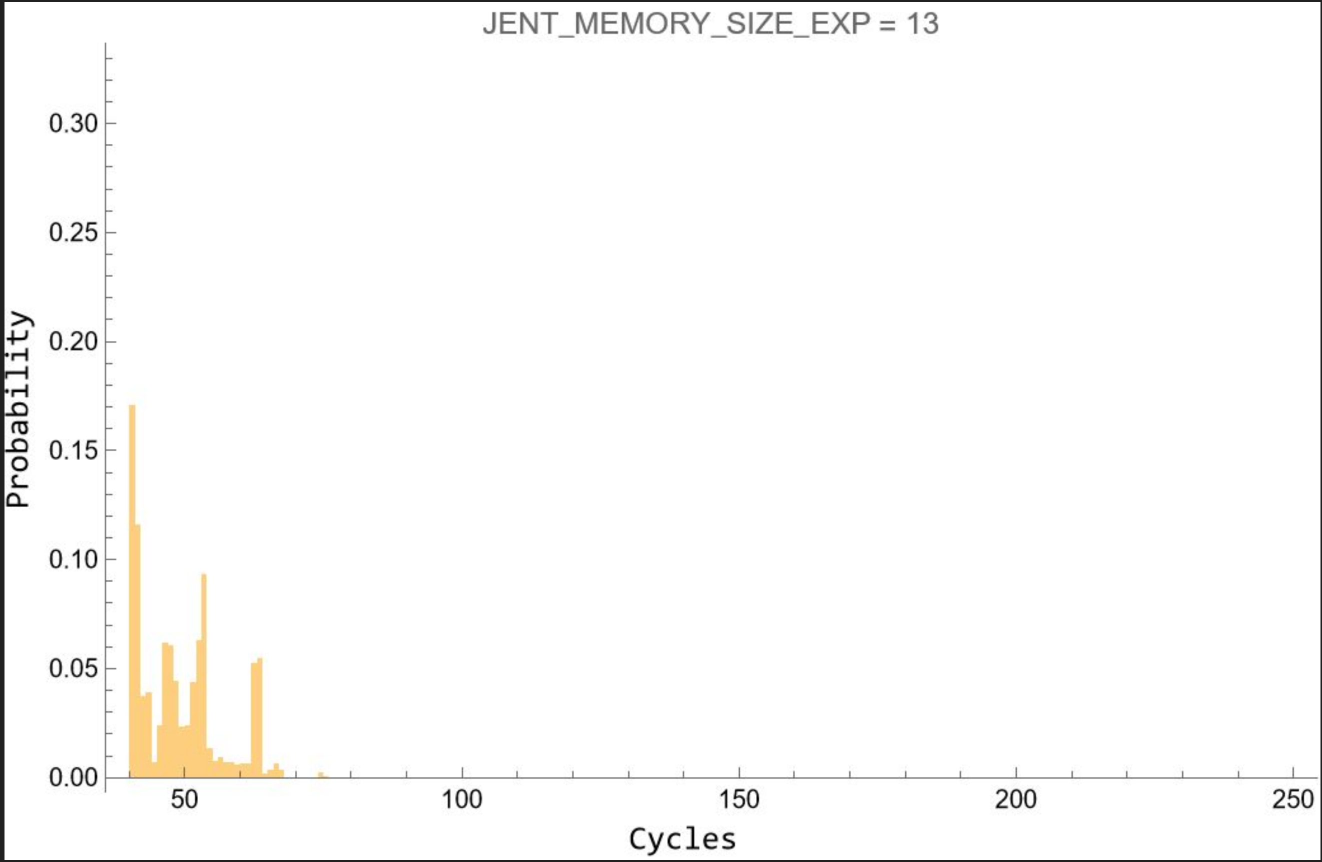
- Once you are decimating sufficiently, you can estimate H_submitter in the obvious way.
- It is probably best (and likely required) that you still don’t make an overall IID claim in the SP 800-90B assessment.
    - There is no general-purpose design-oriented reason this ought to be an IID source!

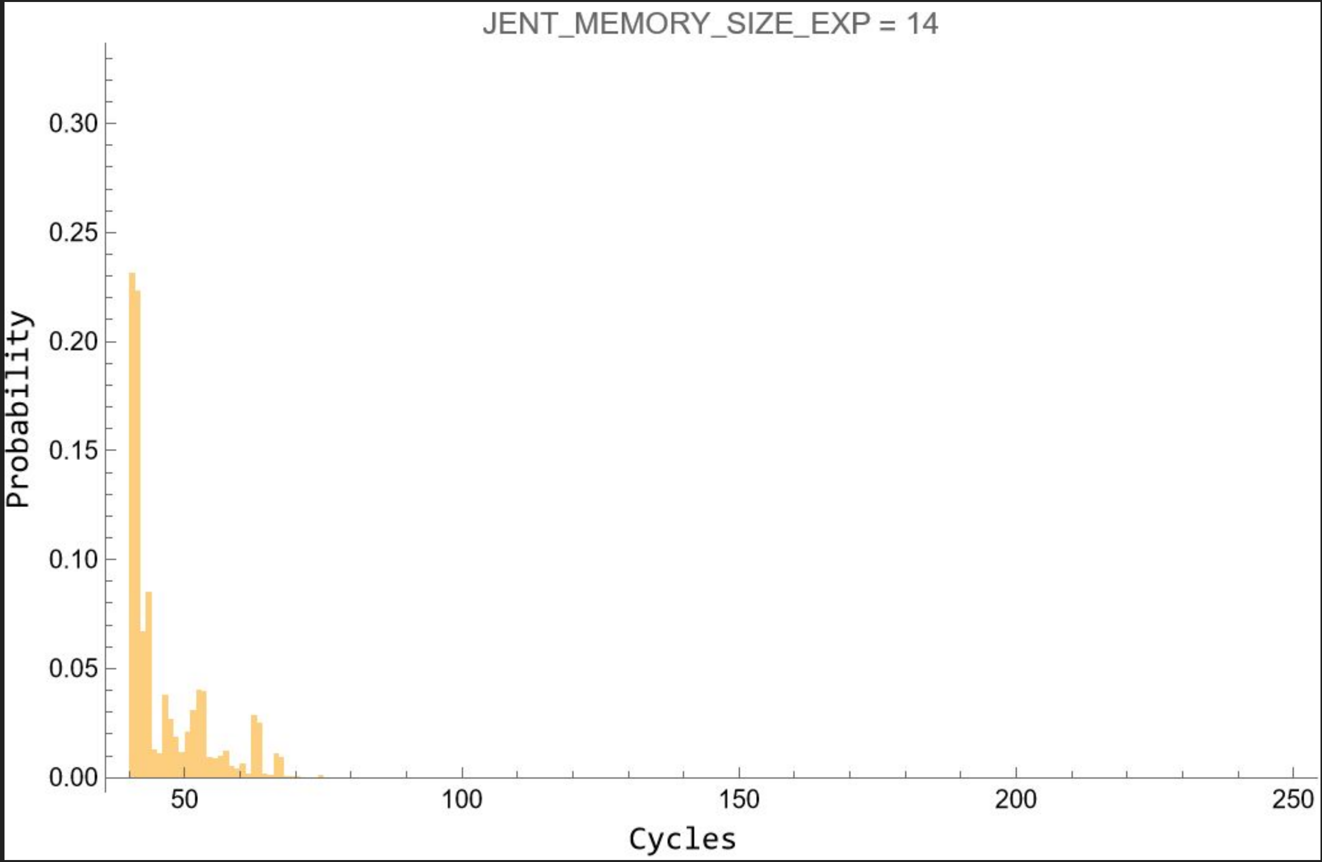
A Worked Example

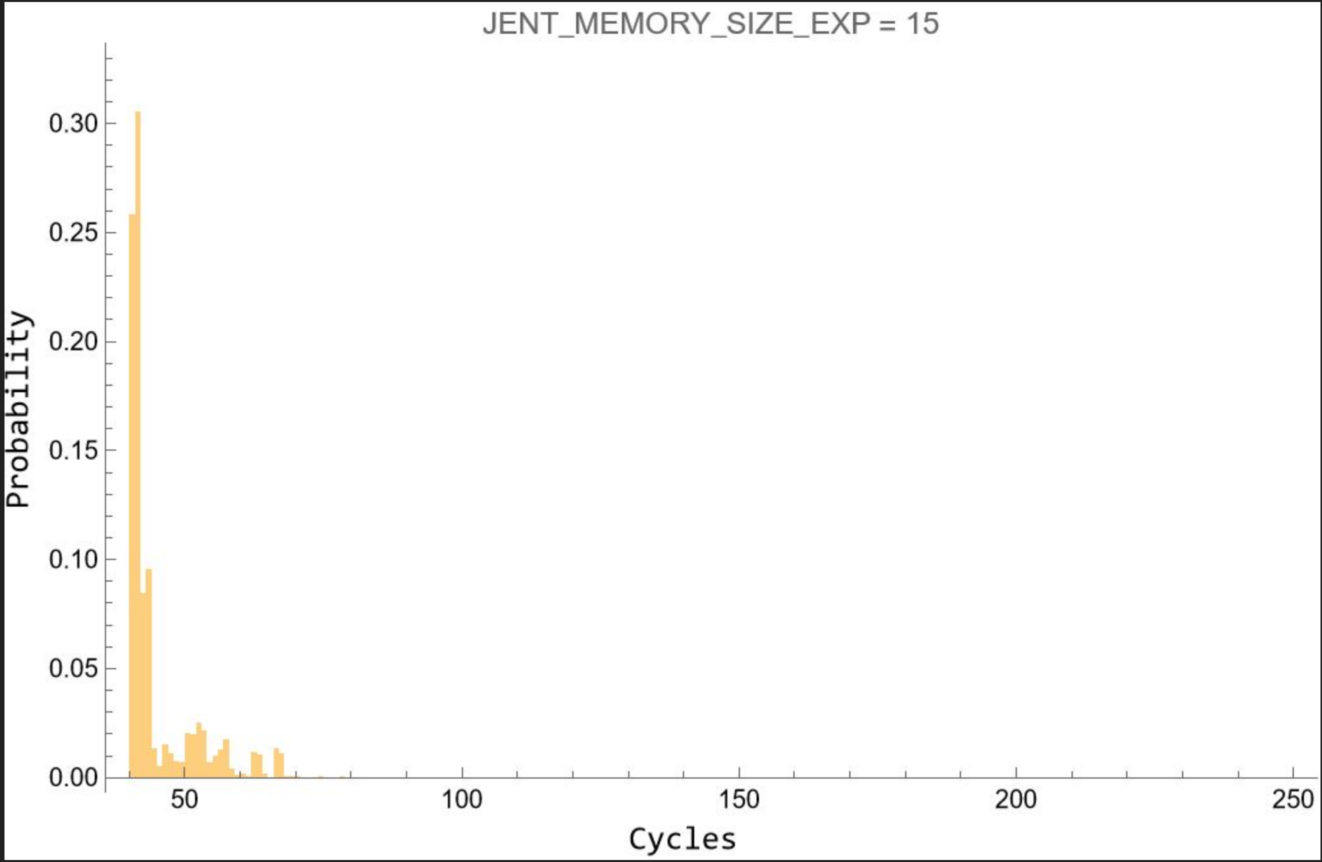

JENT_MEMORY_SIZE_EXP = 10
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles

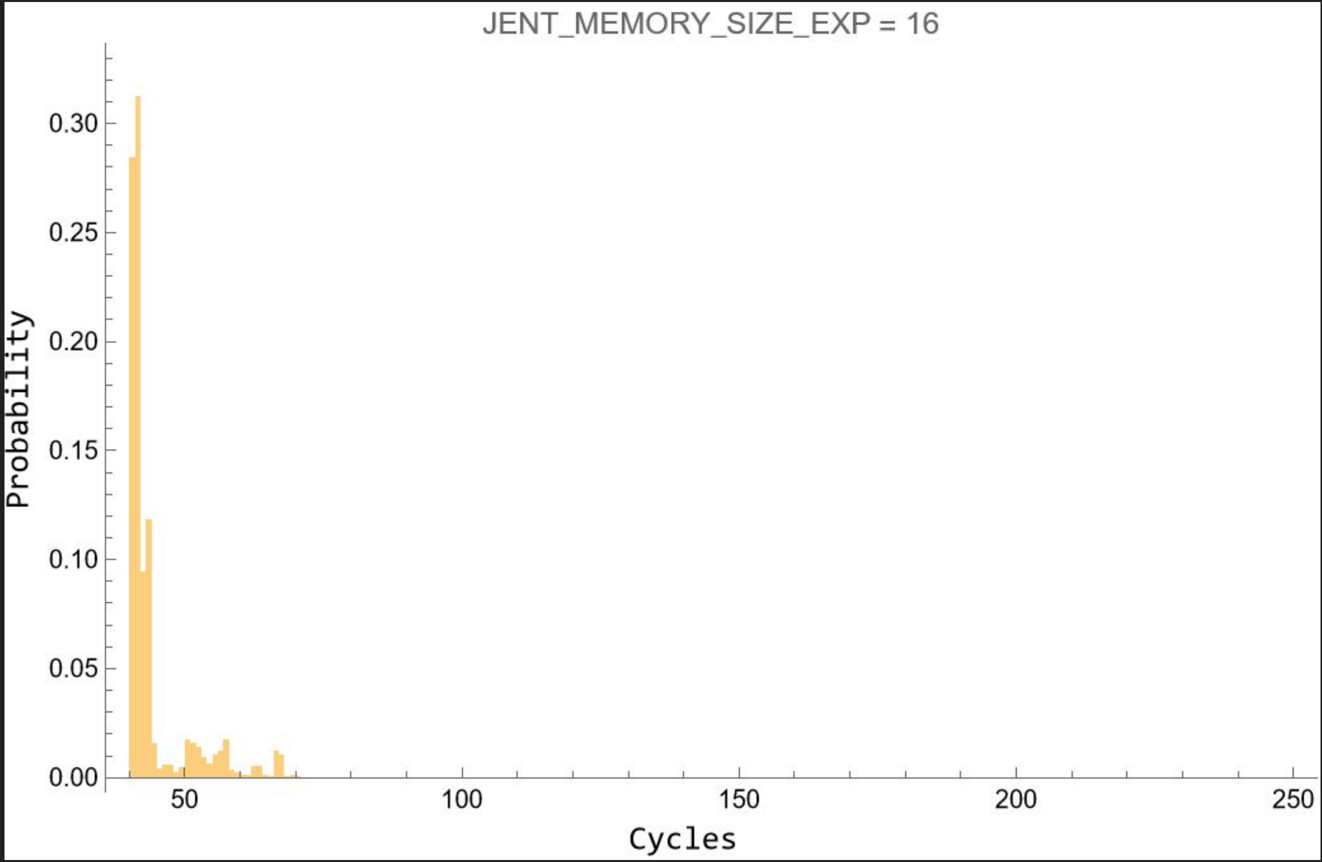

JENT_MEMORY_SIZE_EXP = 11
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles

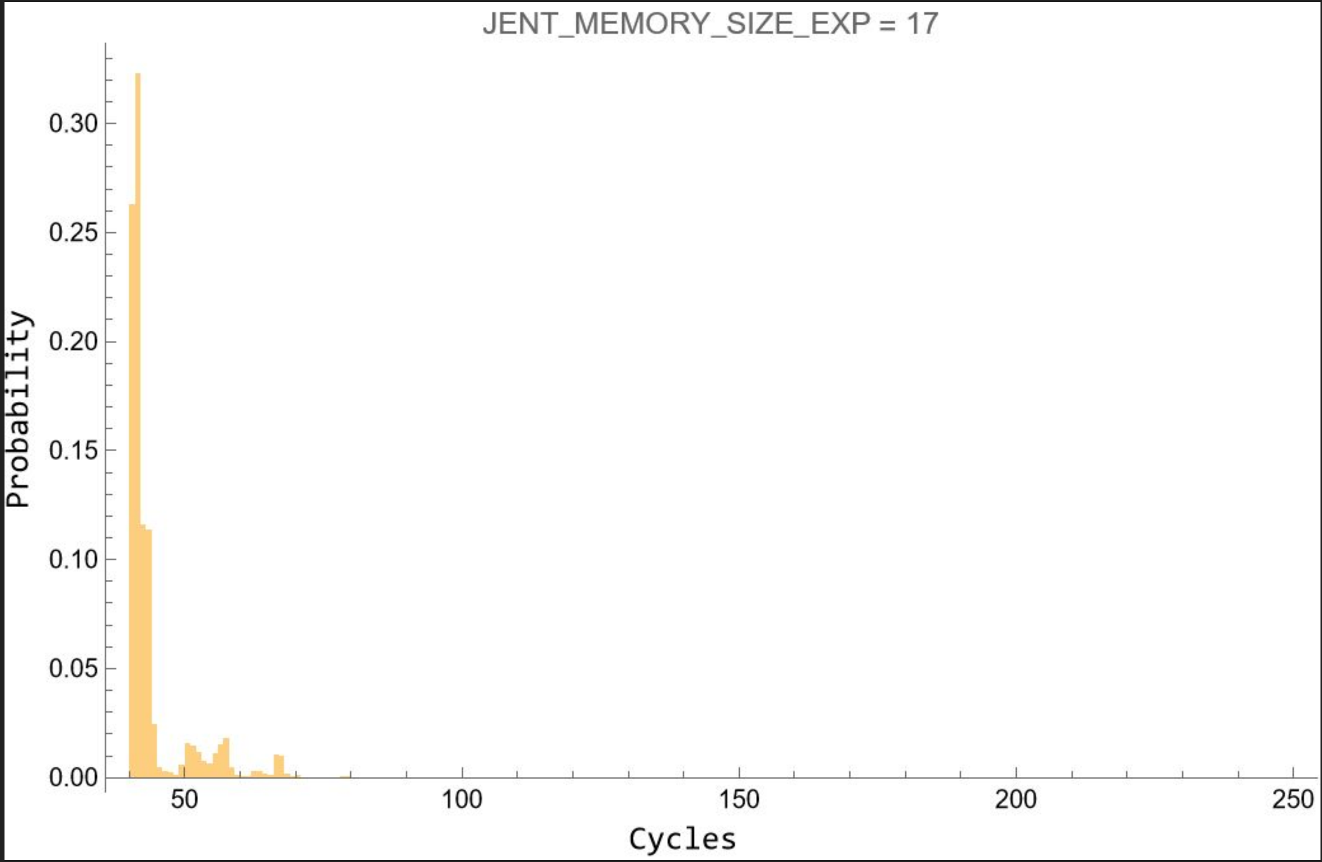

JENT_MEMORY_SIZE_EXP = 12
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles

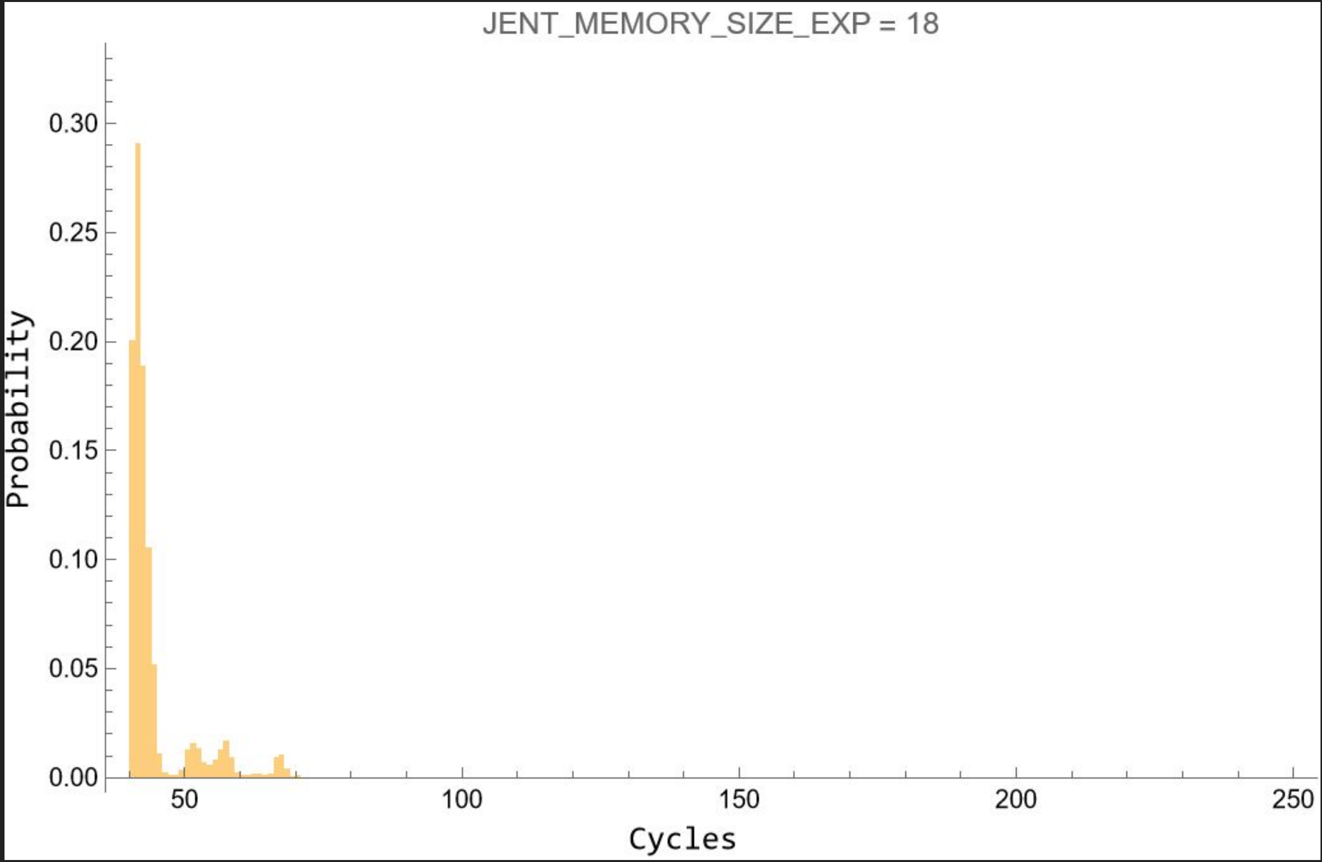

JENT_MEMORY_SIZE_EXP = 13
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles


JENT_MEMORY_SIZE_EXP = 14
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles

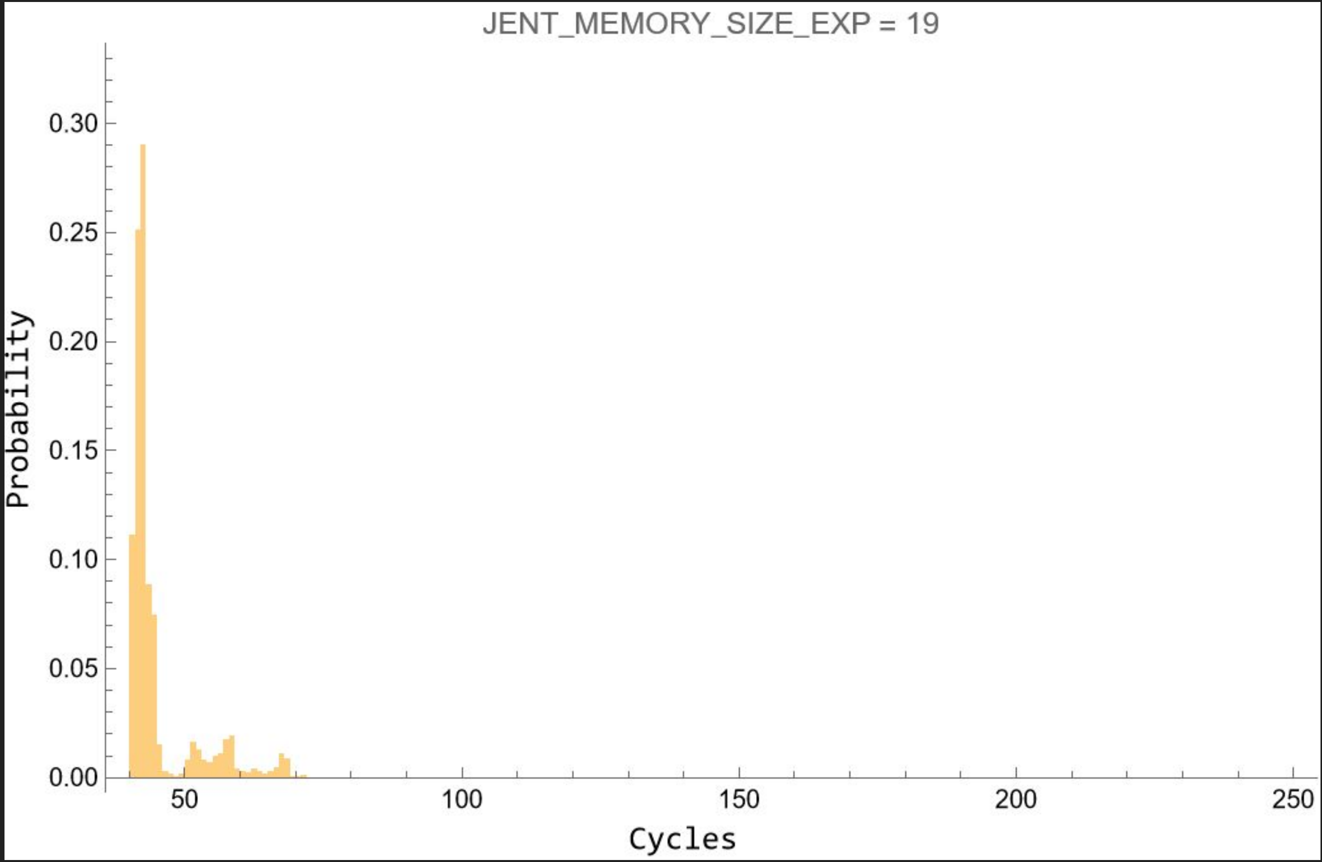

JENT_MEMORY_SIZE_EXP = 15
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles


JENT_MEMORY_SIZE_EXP = 16
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles

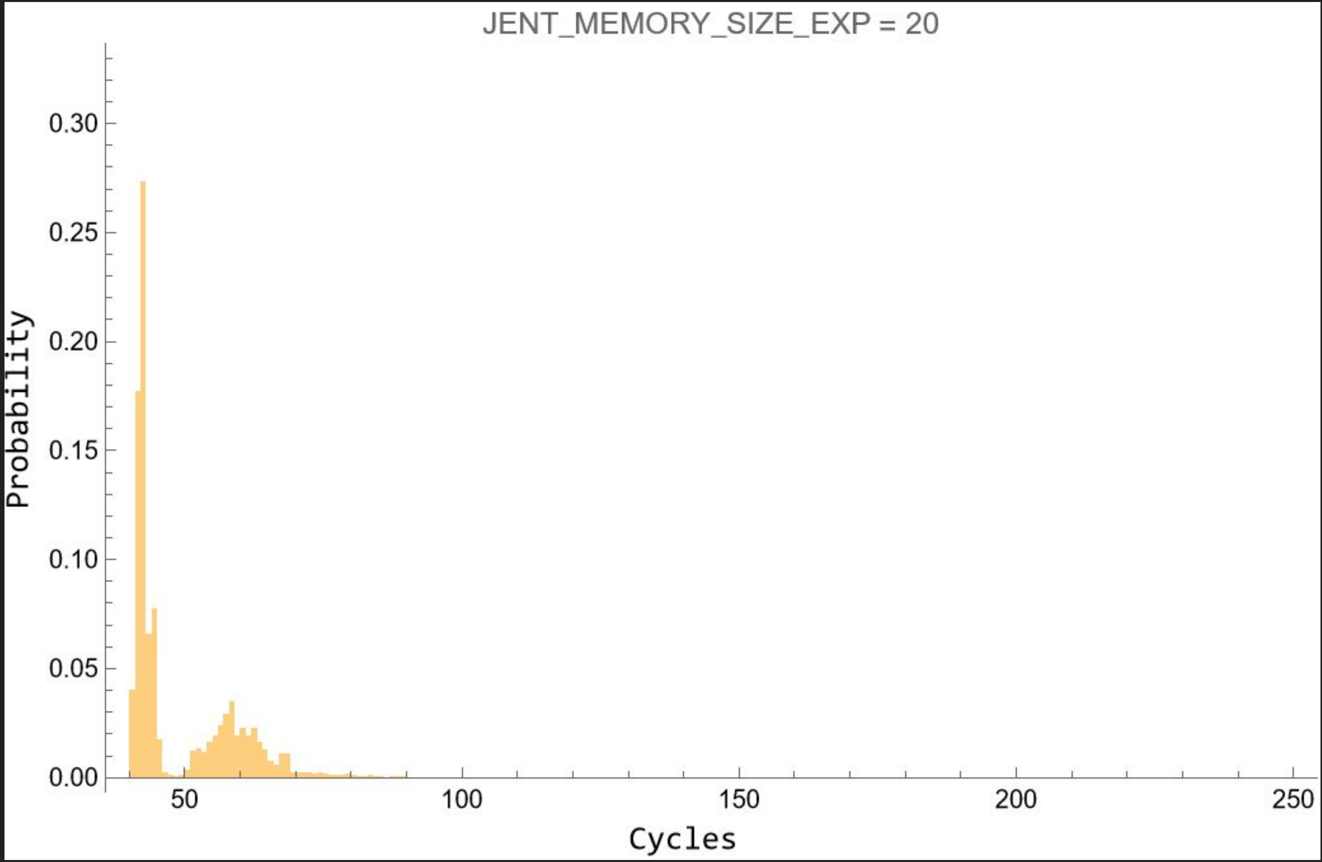

JENT_MEMORY_SIZE_EXP = 17
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles



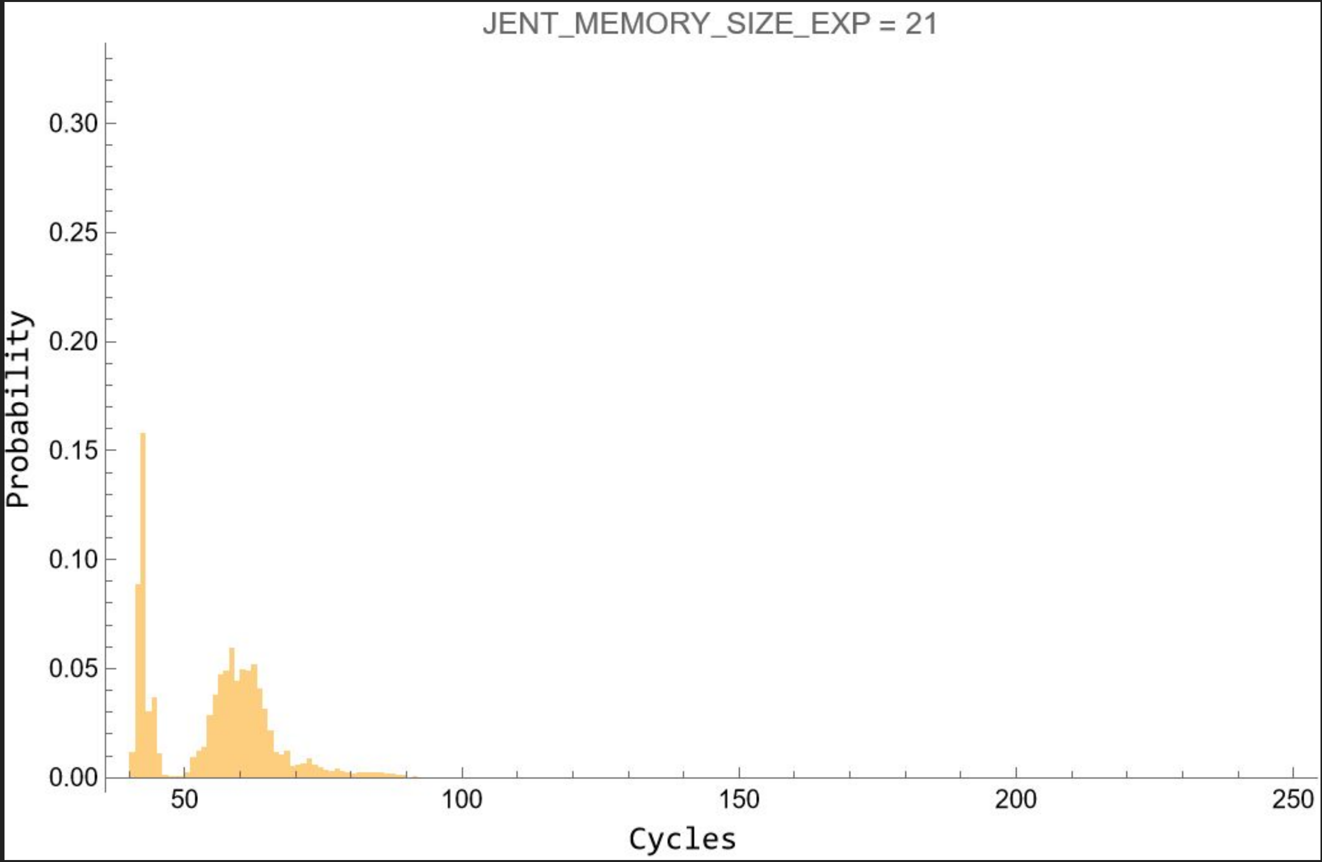
JENT_MEMORY_SIZE_EXP = 18
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles

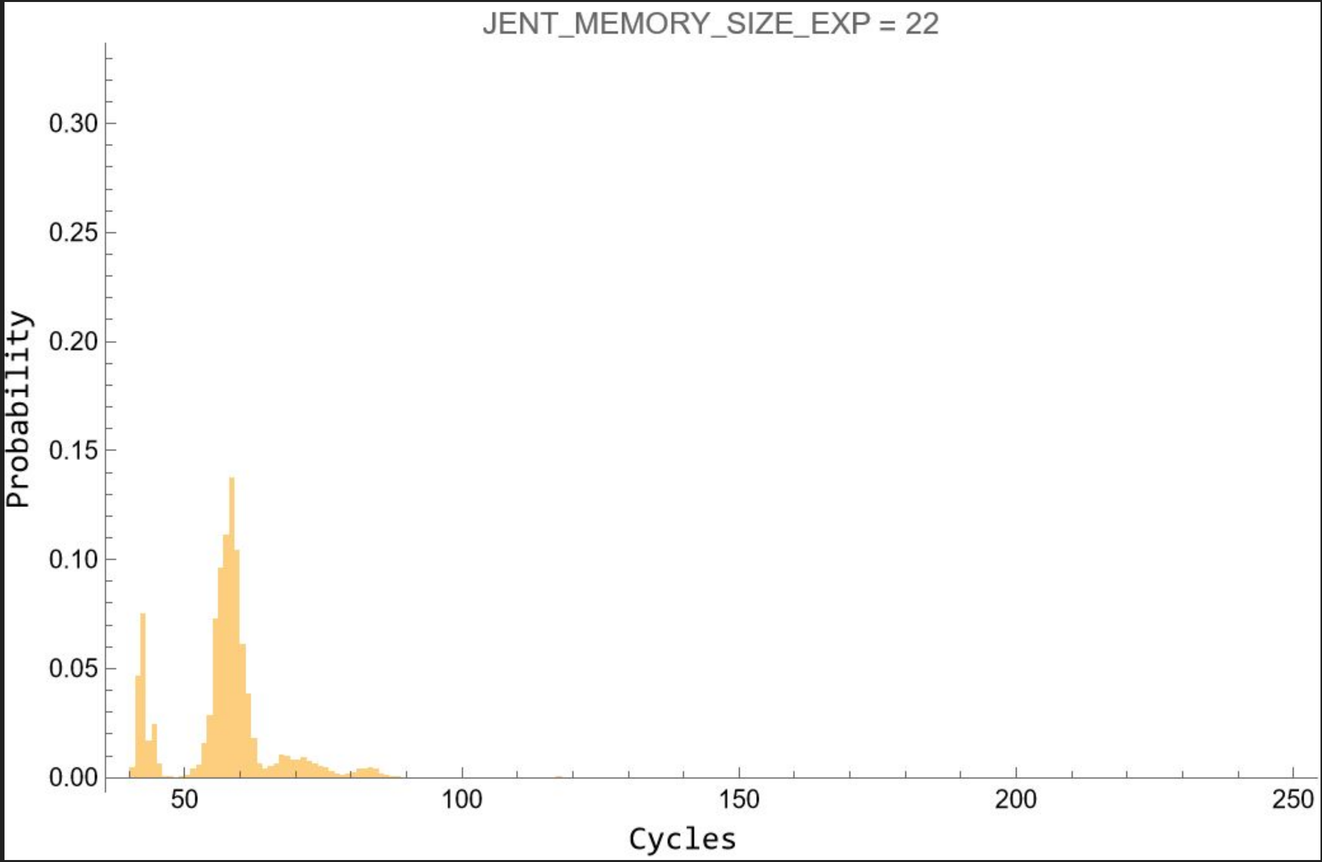


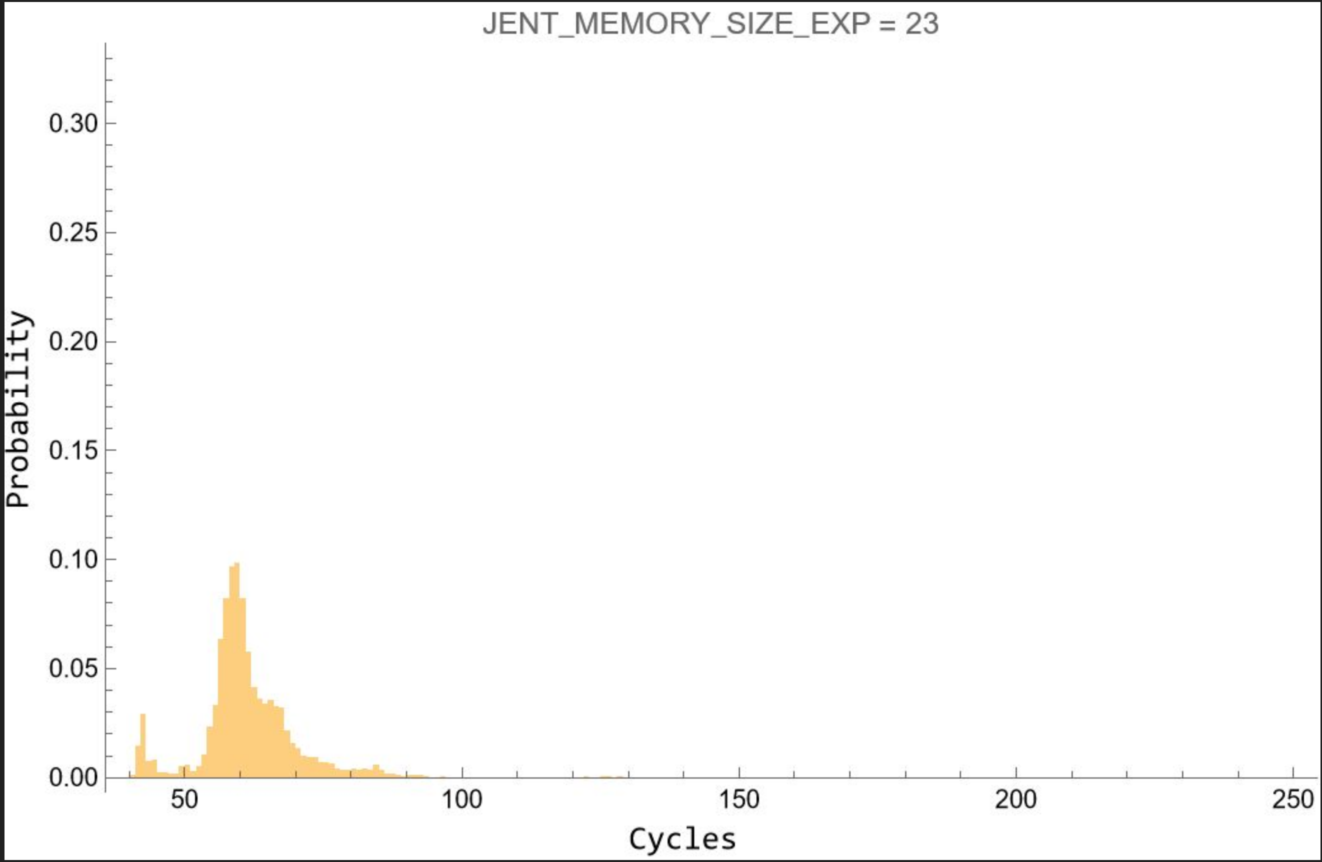
JENT_MEMORY_SIZE_EXP = 19
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles

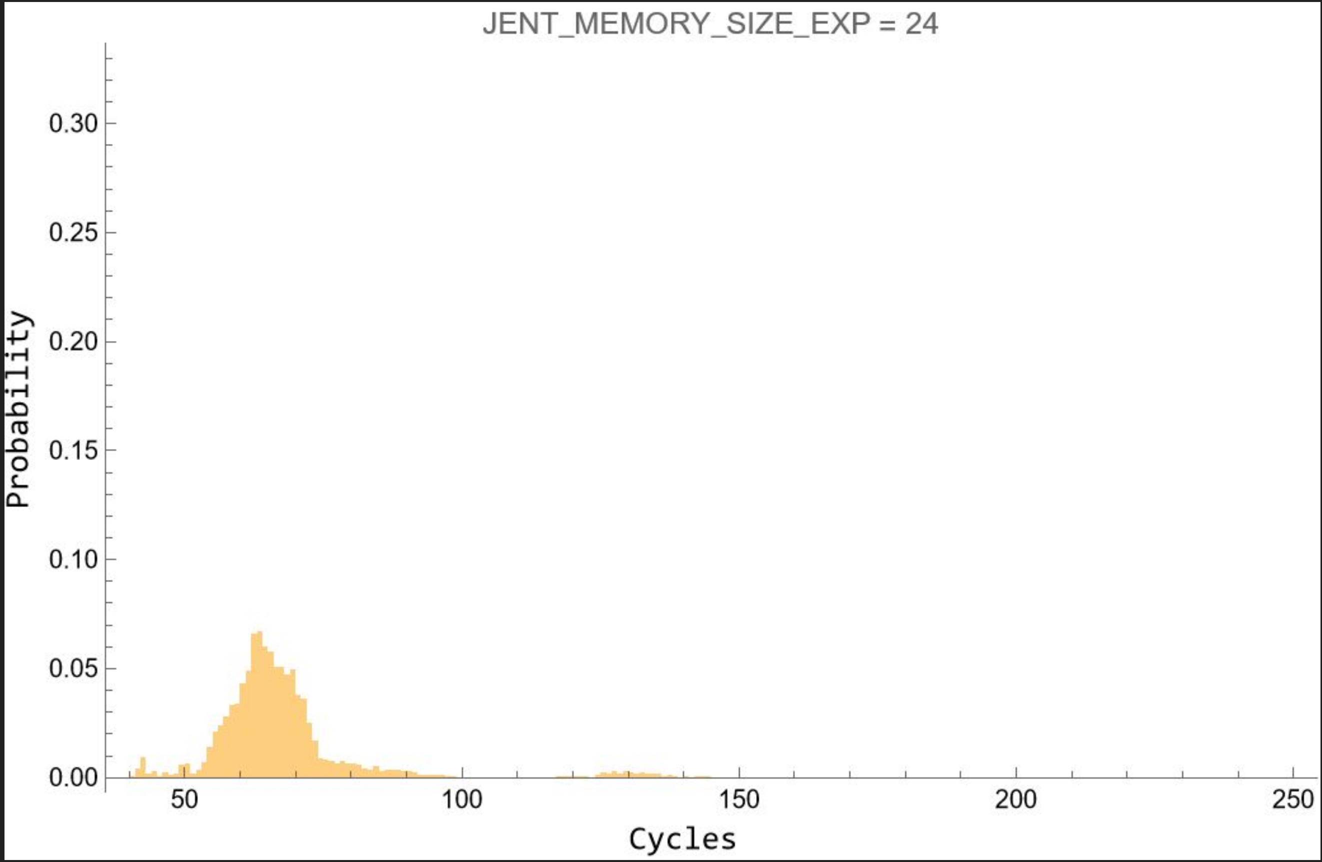


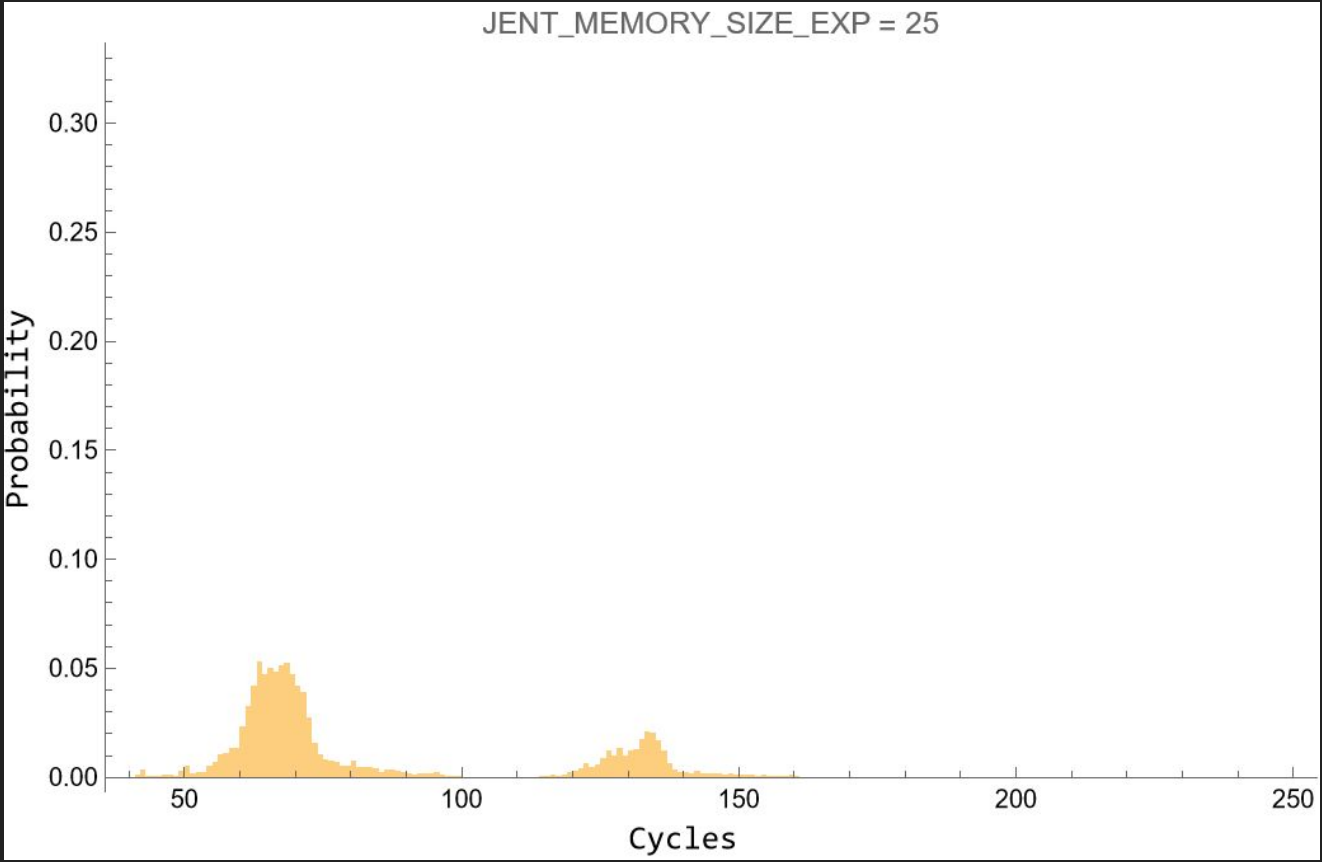
JENT_MEMORY_SIZE_EXP = 20
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles

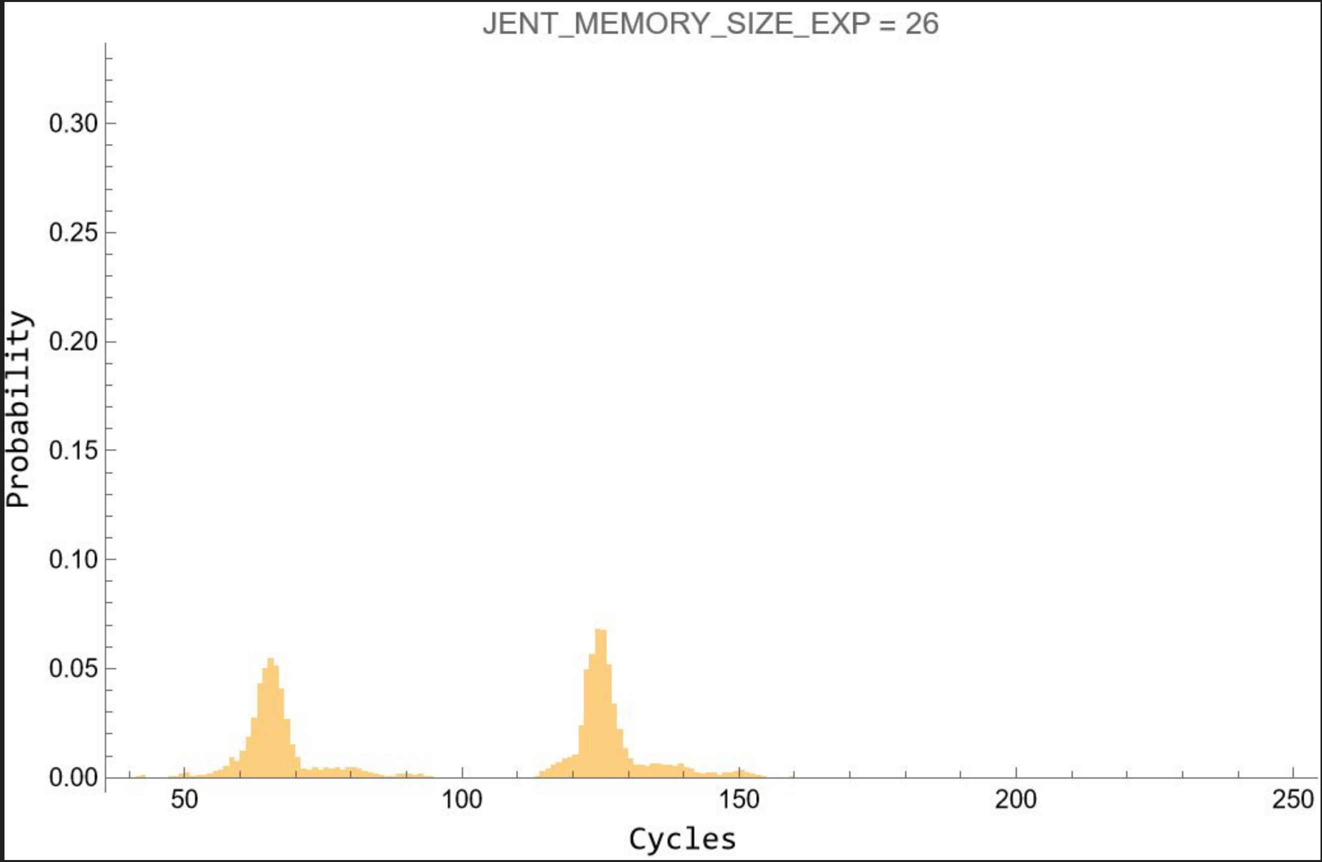

JENT_MEMORY_SIZE_EXP = 21
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles

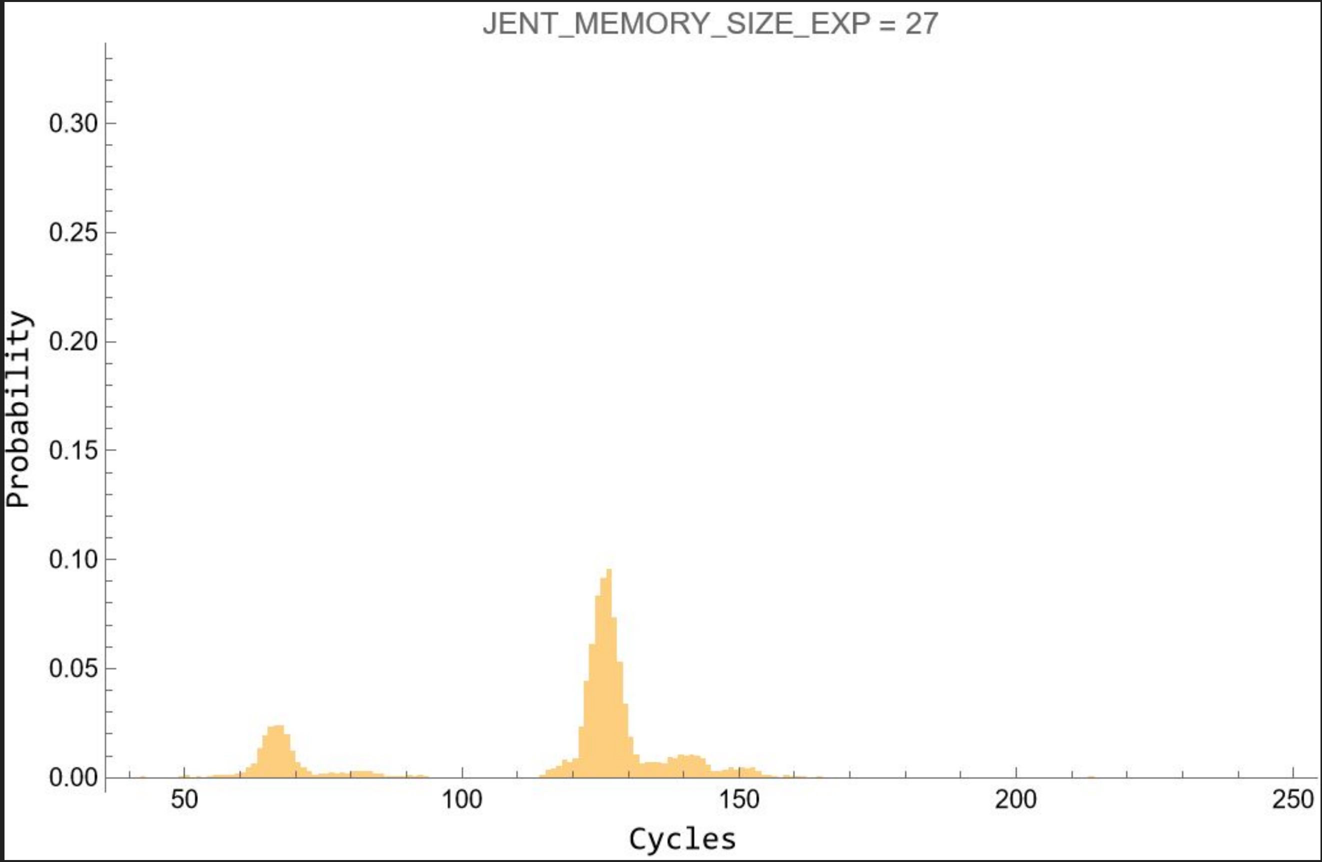

JENT_MEMORY_SIZE_EXP = 22
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles


JENT_MEMORY_SIZE_EXP = 23
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles

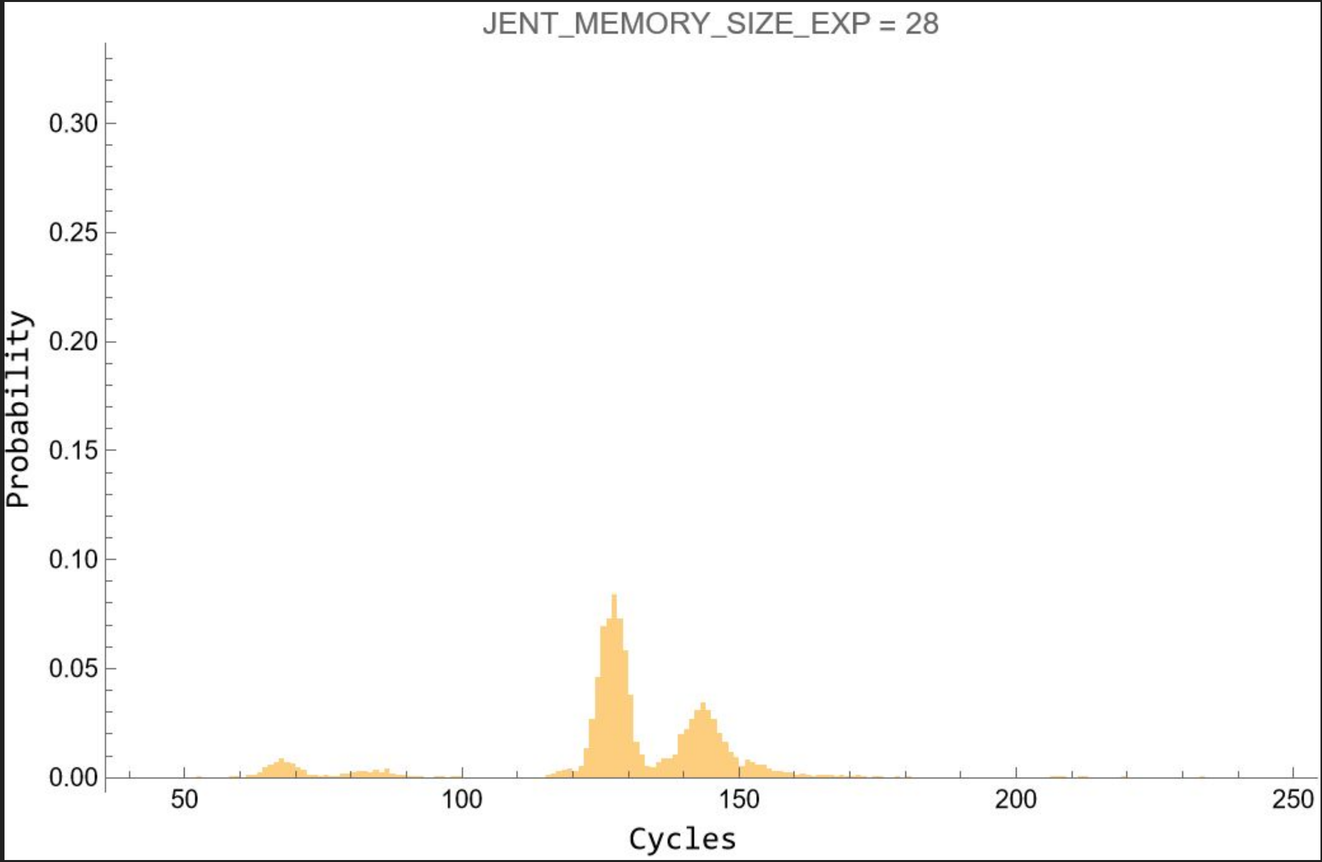

JENT_MEMORY_SIZE_EXP = 24
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles


JENT_MEMORY_SIZE_EXP = 25
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles

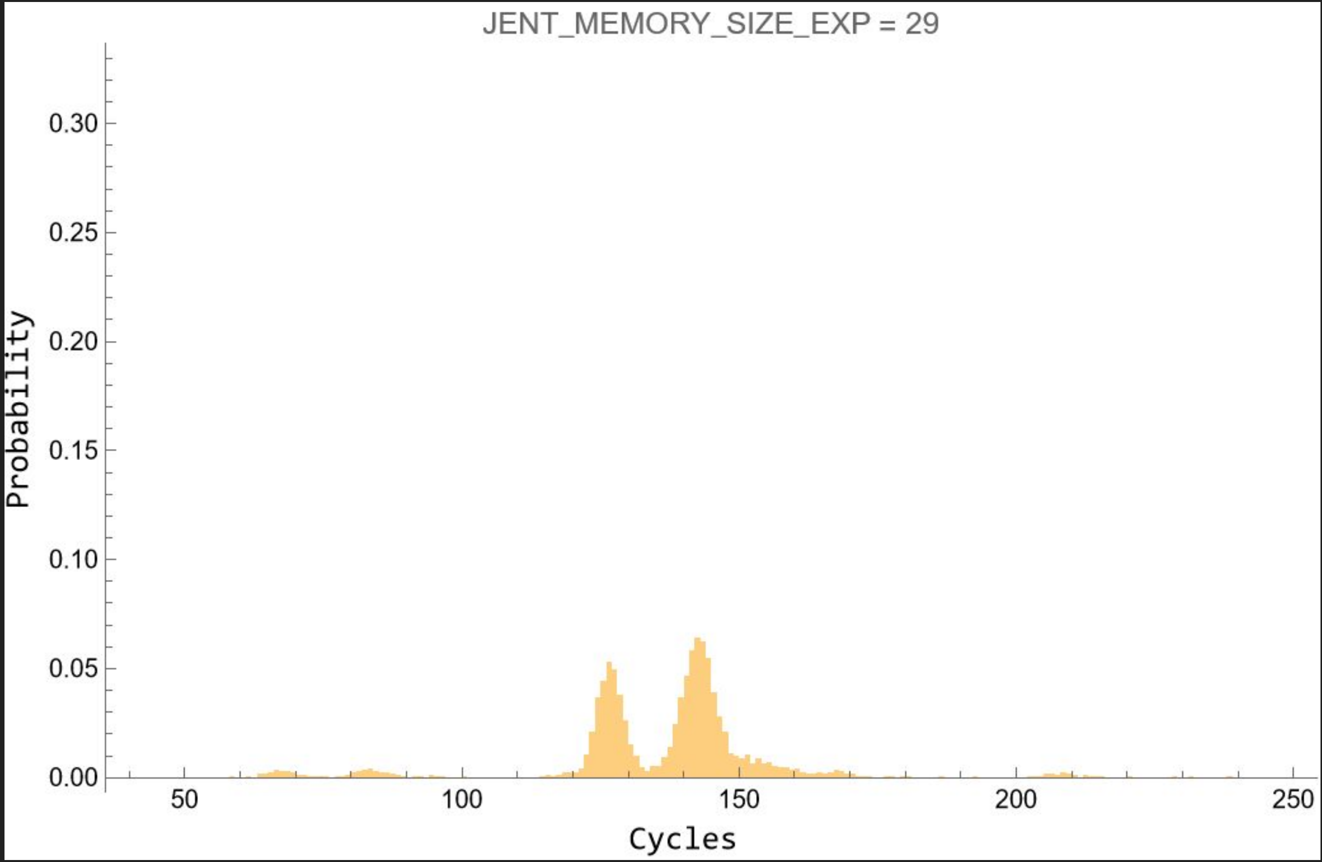

JENT_MEMORY_SIZE_EXP = 26
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles

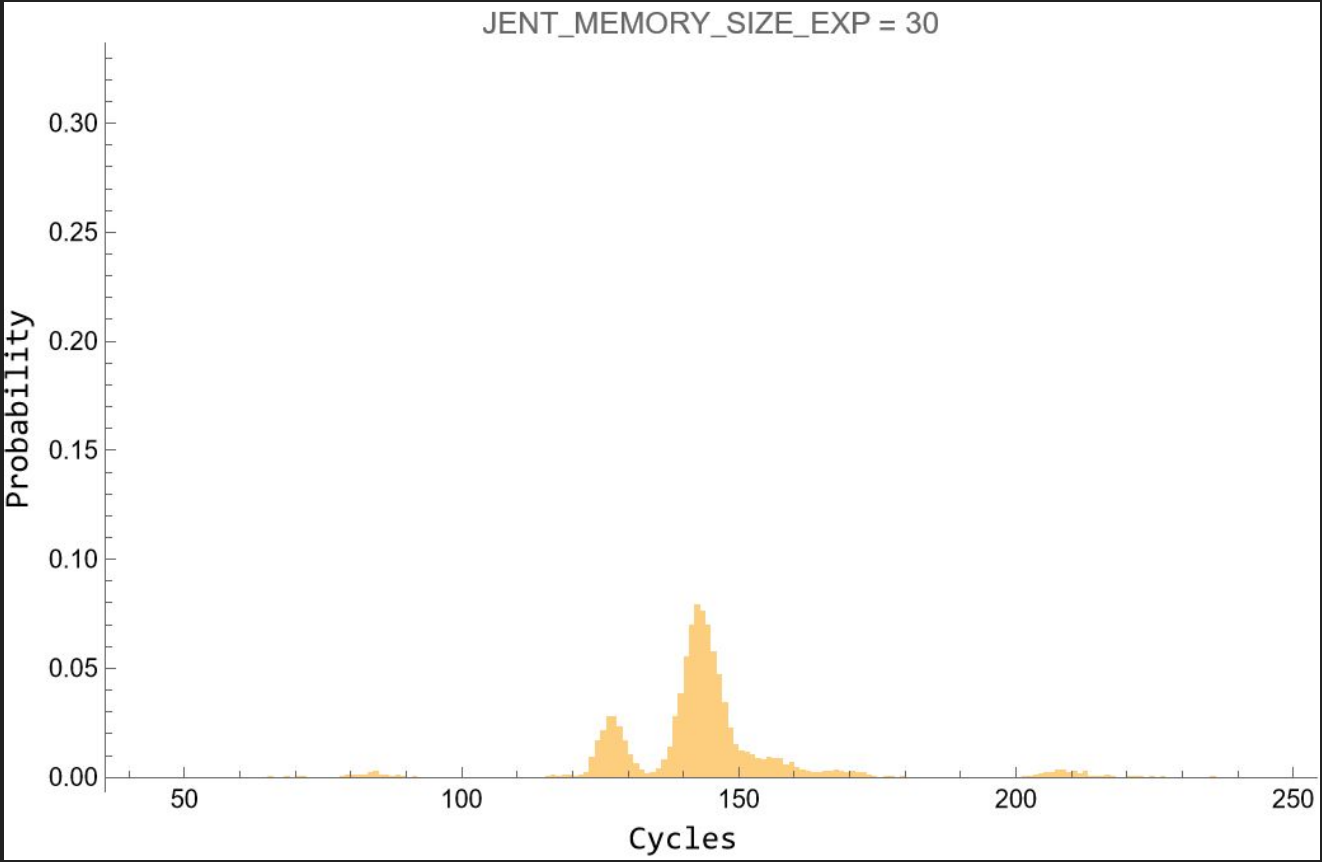


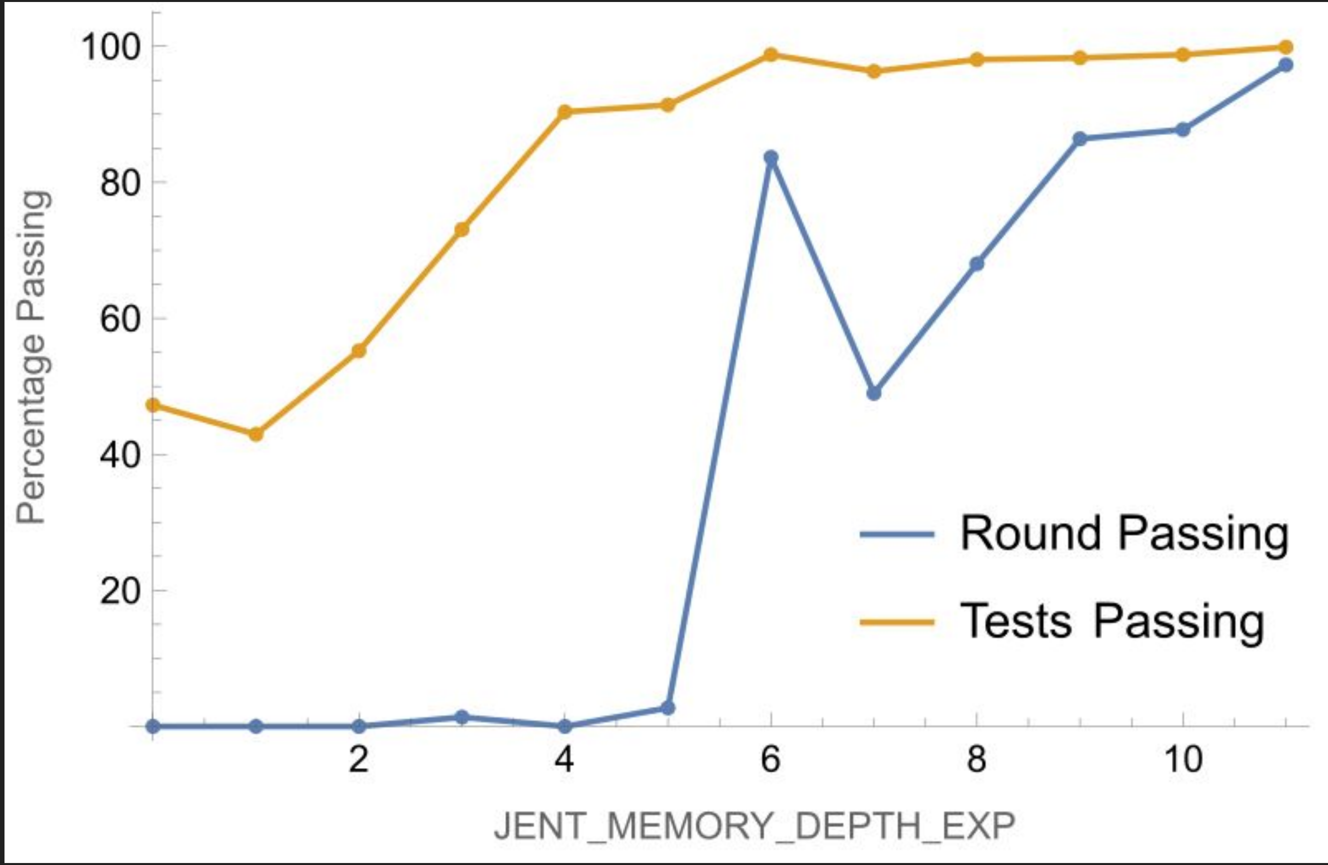
JENT_MEMORY_SIZE_EXP = 27
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles

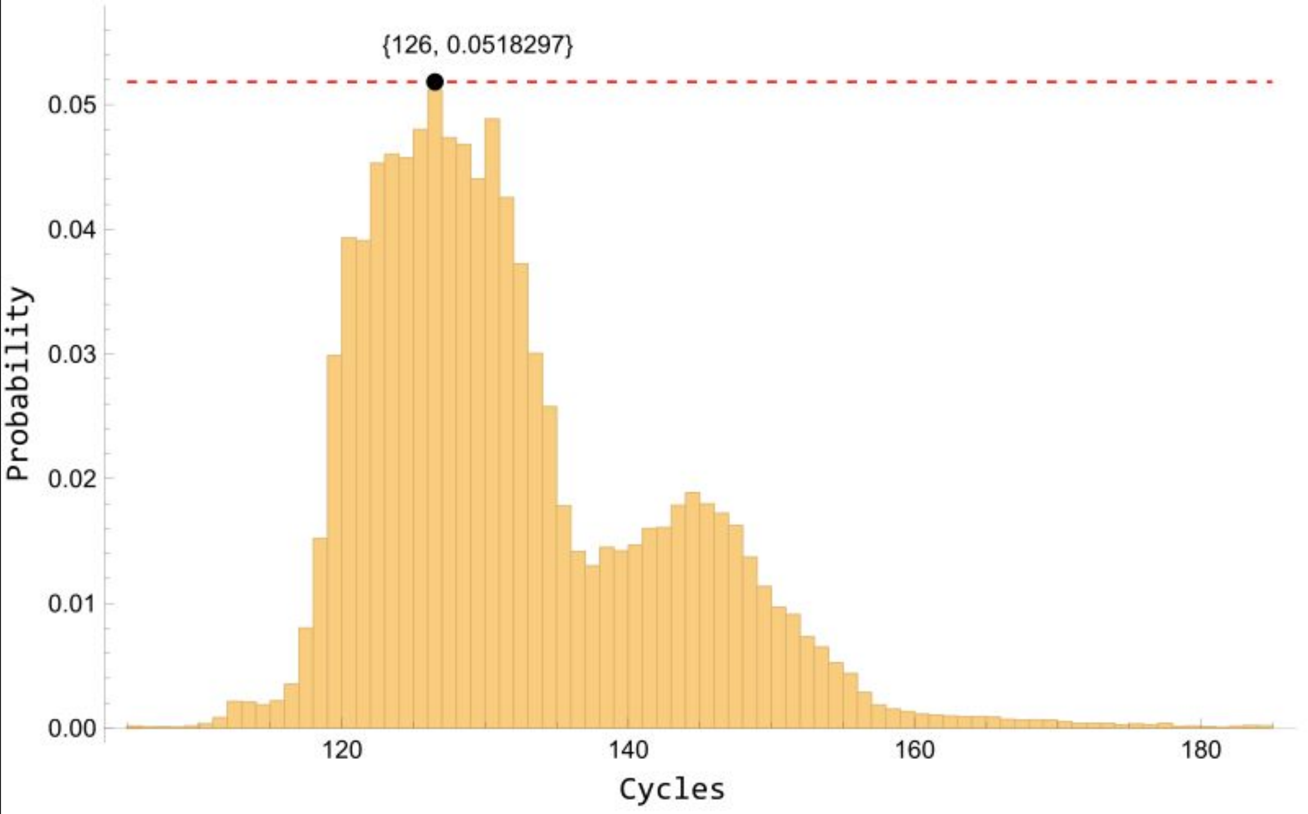


JENT_MEMORY_SIZE_EXP = 28
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles


JENT_MEMORY_SIZE_EXP = 29
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles


JENT_MEMORY_SIZE_EXP = 30
Probability
0.30
0.25
0.20
0.15
0.10
0.05
0.00
50
100
150
200
250
Cycles


100
80
60
40
20
Percentage Passing
2
4
6
8
10
JENT_MEMORY_DEPTH_EXP
Round Passing
Tests Passing



H_submitter = - log_2 (0.0518297) = 4.27

References

- [90B] Meltem Sönmez Turan, Elaine Barker, John Kelsey, Kerry A. McKay, Mary L. Baish and Mike Boyle. Recommendation for the Entropy Sources Used for Random Bit Generation. January 2018.
- [IG] Implementation Guidance for FIPS 140-3 and the Cryptographic Module Validation Program. May 16, 2022.
- [JEnt (MemOnly)] Jitterentropy library with MemOnly updates. https://github.com/joshuaehill/jitterentropy-library/tree/MemOnly and https://github.com/joshuaehill/jitterentropy-library/tree/MemOnlyPR
- [JEnt (Original)] Jitterentropy library. https://github.com/smuellerDD/jitterentropy-library
- Jitter RNG SP800-90B Entropy Analysis Tool. https://github.com/joshuaehill/jitterentropy-library/blob/MemOnly/tests/raw-entropy/README.md
- [Müller 2022] Stephan Müller, CPU Time Jitter Based Non-Physical True Random Number Generator, July 1, 2022.

Thank you!

KeyPair CONSULTING