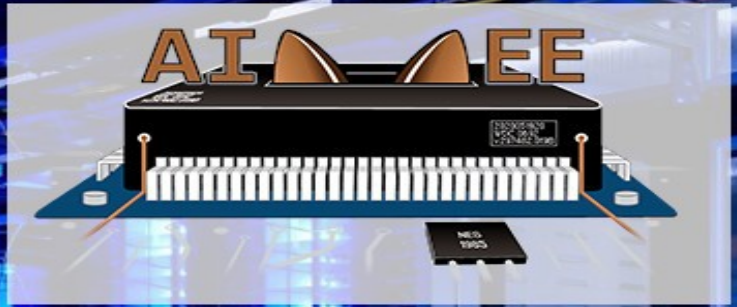
Evolving ROP Chains

Francisco Blas Izquierdo Riera,
Lucca Fraser & Lauren Moos
Special Circumstances LLC

History of Return Oriented Programming

History of ROP

- At the beginning stacks were executable
- Buffer overflow attacks were simple:
  1) Overwrite return address to point to stack
  2) Follow return address with code to be ran
- Stack position was randomized (ASLR)
  - Increase the injected program size adding No Operation (NOP) instructions to make falling somewhere in those instructions likely
- Stack was made non executable
  - Find already executable code that could be useful (gadgets)
  - Make them return in to each other to attain desired result (ROP)



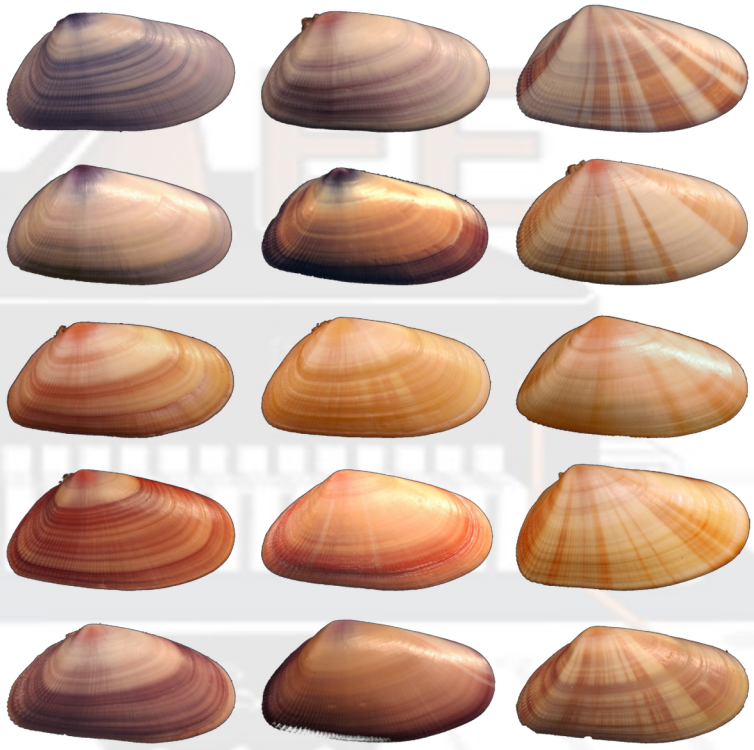
Genetic Programming Design Decisions

Genotypes: Two Approaches

1) Vector of machine words
   - Each genotype a potential ROP payload
   - Phenotype developed by
     - Writing genotype to the stack of the (emulated) target process
     - Setting the IP to the head
     - Incrementing the SP
     - Executing emulation
2) Vector of instructions intermediary, payload-constructing VM
   - Programs aware of memory image
   - Semantic analysis primitives available
   - Outputs a payload of type (1)...

Phenotype

- Execution behaviour monitored through Unicorn hooks and callbacks
- Attributes are committed to record only when the execution reaches a composable point – a return, for example.
- A variety of attributes are recorded
- A fitness formula, with these attributes for variables, is given by experiment configuration file
- Fitness scalar obtained by evaluated fitness formula



Selection

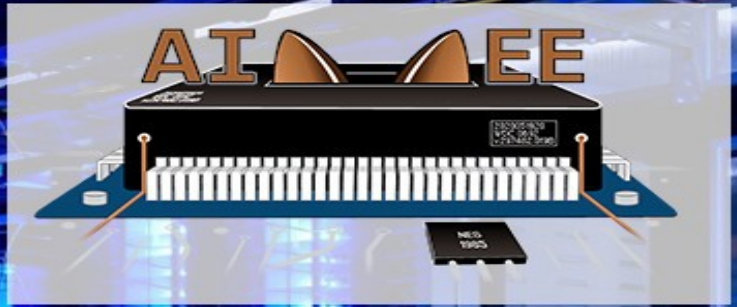
- Different methodologies tested:
  - Roulette
  - Pareto front
  - Lexicase
  - Tournament selection
- Tournament selection performed best
  - Optional geography
  - N (usually 5) contestants drawn on each iteration and evaluated
  - P (usually 2) best are bred
  - P offspring displace the P worst performers

Geography

1) Islands with migration
   - Simple parallelization
   - Fosters diversity
   - Each island is handled independently
   - Each island may send or receive individuals through a shared “pier” (lockfree queue) randomly
2) Linear geographies
   - Circular buffer on each island
   - First contestant picked randomly
   - Other contestants picked from neighbors within n slots
     - If n is the buffer size selection is unrestricted

Genetic operators

- Alternating crossover
  - Stitch together from alternating between parents
  - Length from exp. dist.
  - Prevents genetic bloat (maximum child length = maximum parent length)
- Single point crossover
  - Pick two cut points and concatenate results
- Mutation
  - Integer bitwise operators
  - Replace with memory reference
  - Replace with memory contents



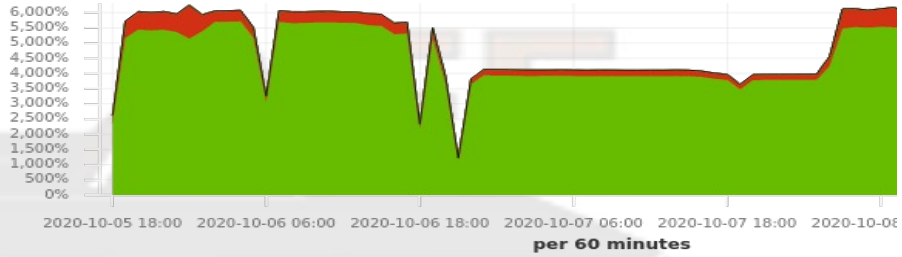
Technical obstacles

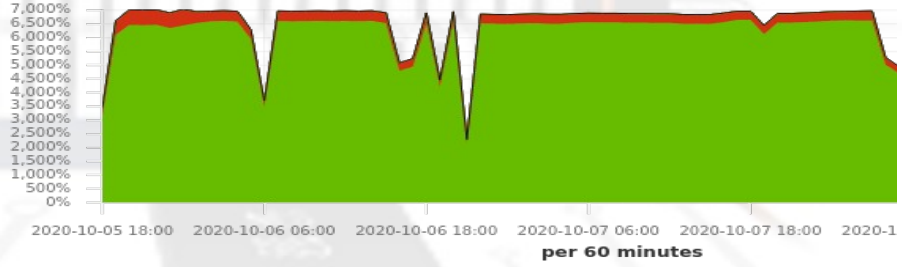
Race condition in unicorn

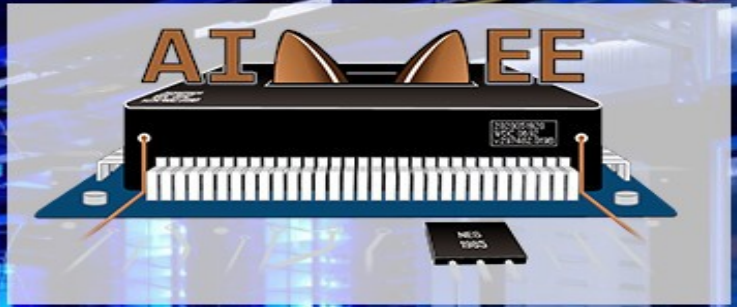
- Infrequent segmentation faults encountered in large-scale experiments
- Traces point to null struct write on unicorn library
- Cause: timeout callbacks racing against main thread
- Solution: synchronize accesses using a mutex

CPU scalability issues

- Benchmarks testing slowed down by the race condition
- Once fixed system won’t scale over 4/5 cores
- But system load shows enough workers were waiting
- Suspected causes are memory or I/O pressure







Experiments

The Effects of Phenotypic Novelty Pressures

Zero Count Experiment

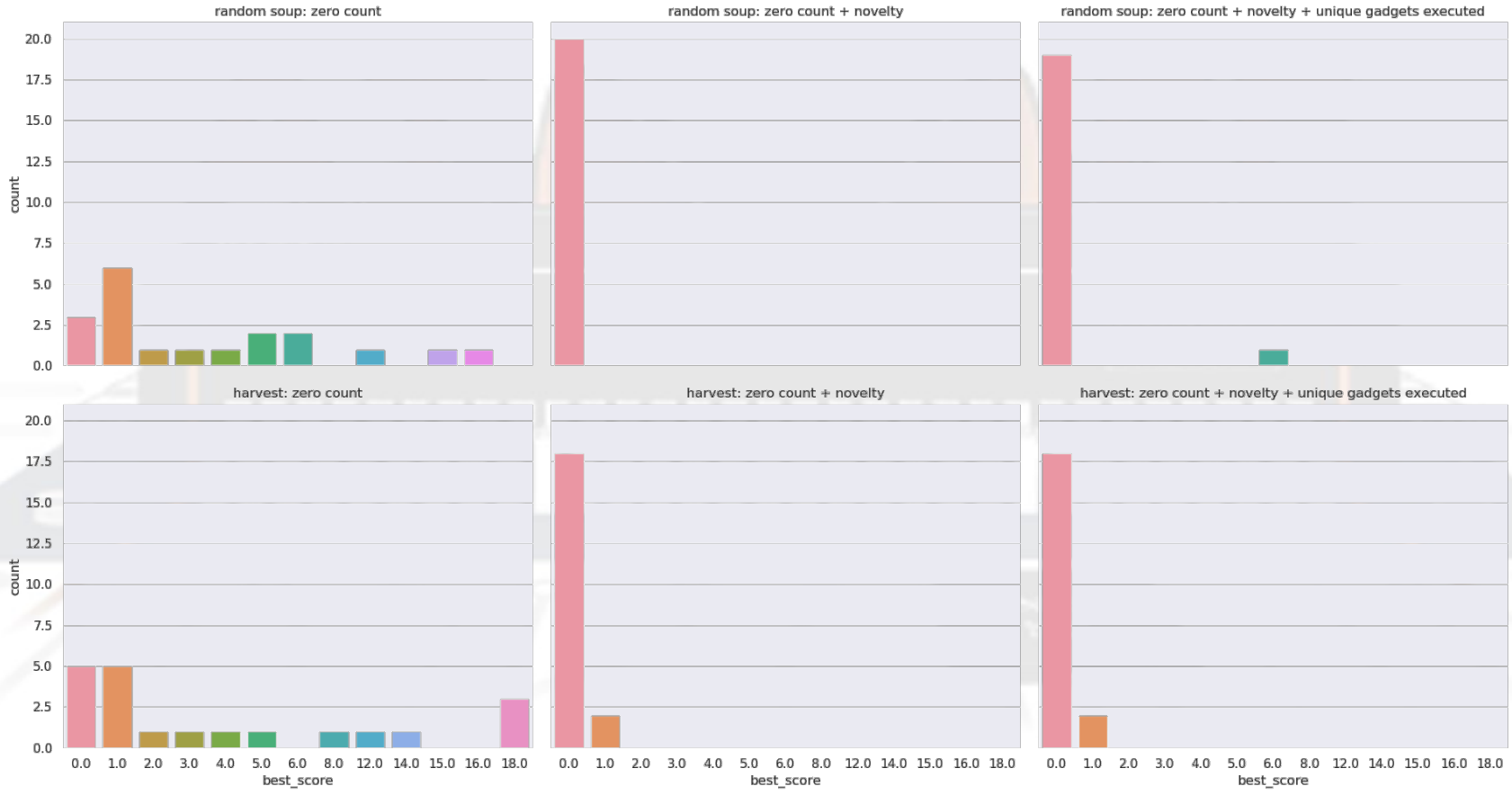
- Objective is to minimize the zero bits in the bitwise product:

  `EAX & EBX & ECX & EDX`

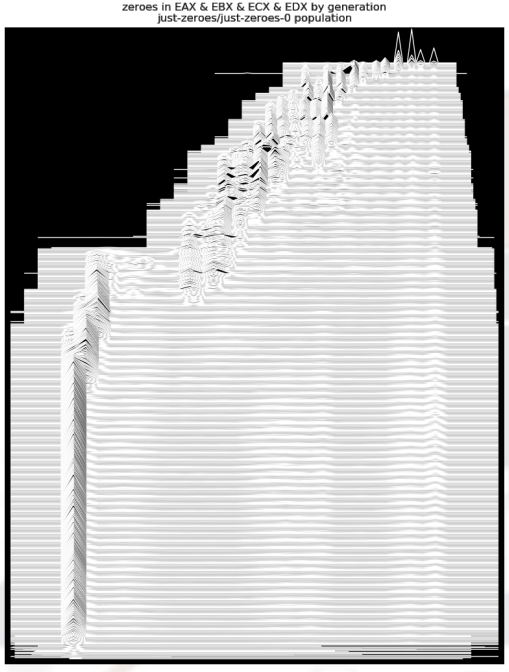
- Target binary is a statically compiled OpenSSH_6.8p1 for i386 architecture

- Phenotypic novelty is measured by feeding resulting register vectors into a CountMinSketch
- We compare runs in which novelty is a component of the fitness score, with runs in which it is not

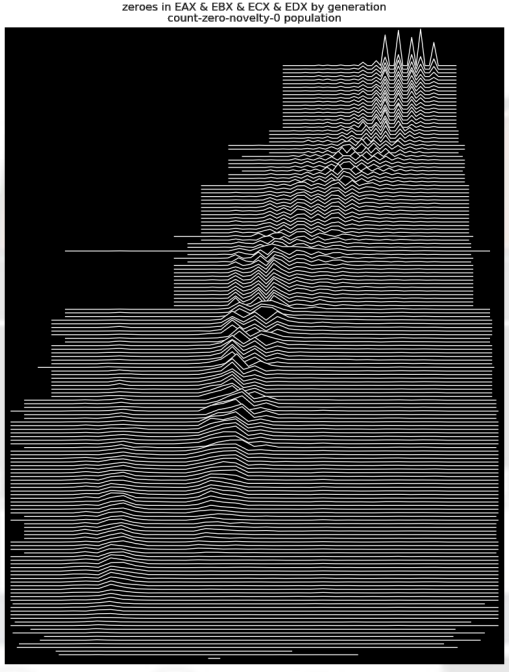
Distribution of Solutions



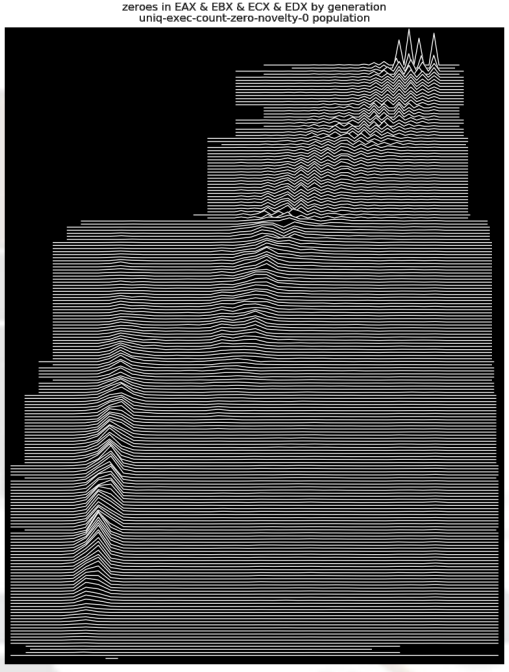
Distribution of Fitness by Generation



No novelty pressure

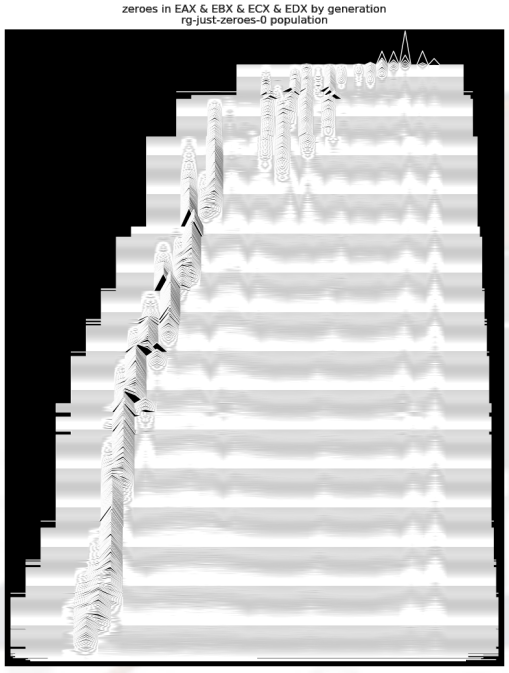


With novelty pressure

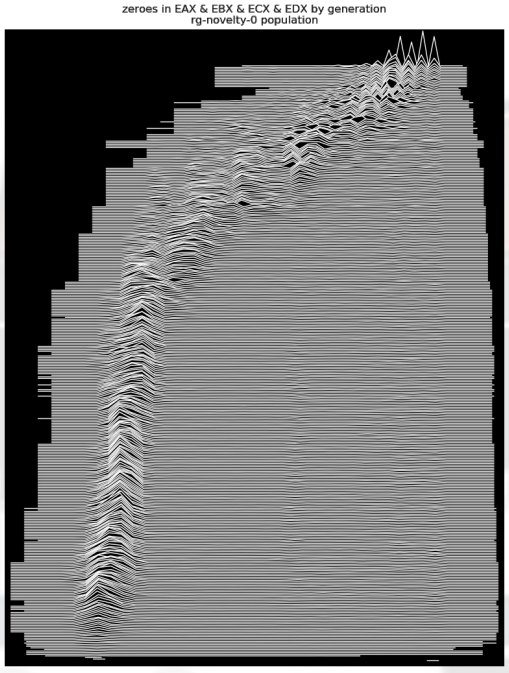


With novelty + unique executed gadget pressure

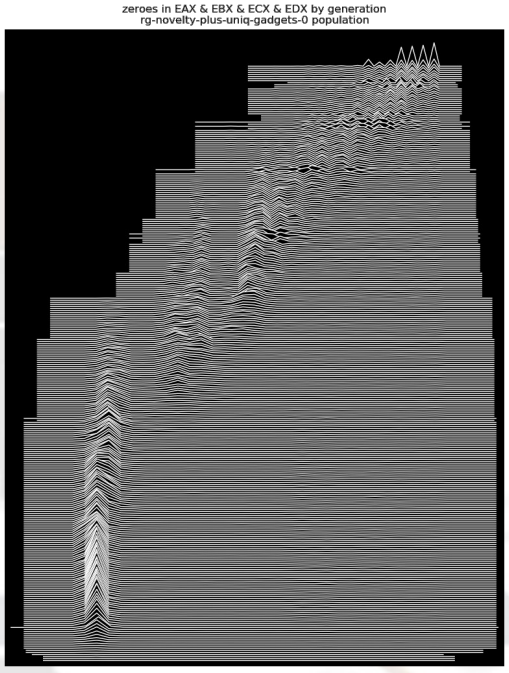
Distribution of Fitness by Generation



No novelty pressure



With novelty pressure



With novelty + unique executed gadget pressure

Sex & Composability

Sexual reproduction and composability

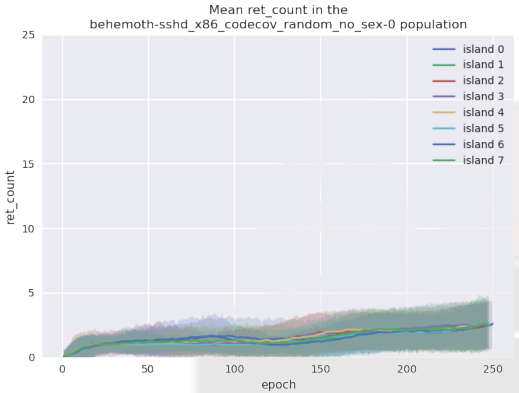
- The mixability theory for the role of sex in evolution holds that sexual reproduction favours the emergence of genes that perform well in multiple genetic contexts.
- Our conjecture: crossover should favour gadgets that **compose well** with others.

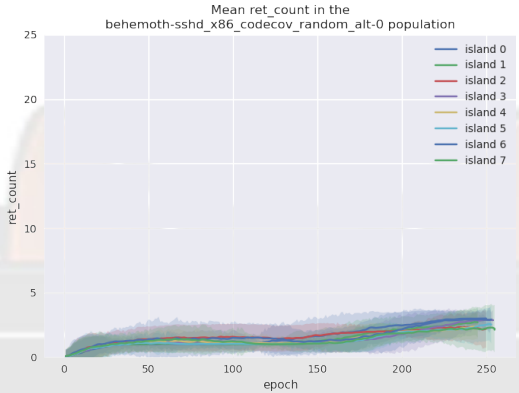
- See: Adi Livnata, Christos Papadimitrioub, Jonathan Dushoffc, & Marcus W. Feldmand, "**A mixability theory for the role of sex in evolution**", PNAS December 16, 2008 105 (50) 19803-19808.

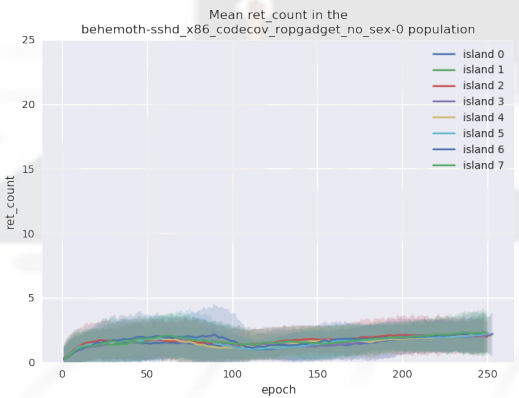
Sexual reproduction and composability

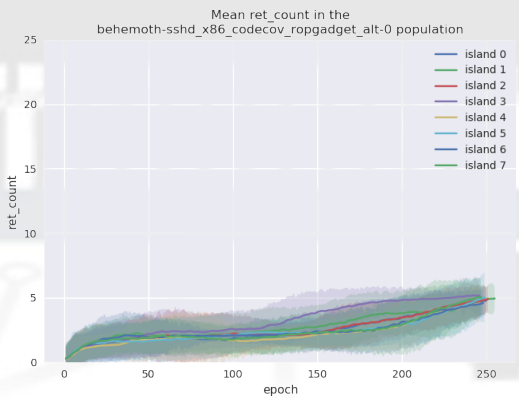
- Tested three crossover approaches:
  - Asexual (only mutations)
  - Alternating
  - Single-Point
- Tested two seeding techniques:
  - Random initial values
  - Values extracted by ROPgadget
- Metrics:
  - Return instruction executed
  - Code coverage
  - Allele circulation
  - Generational distribution
- Fitness function: code coverage

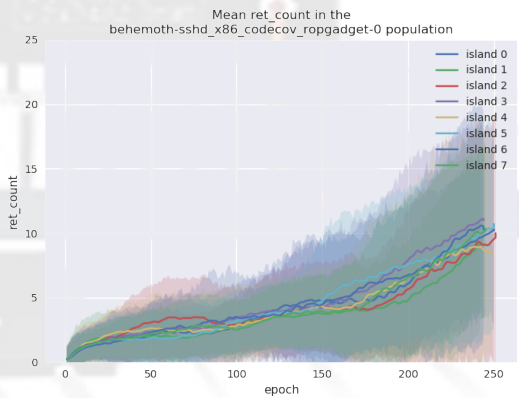
Return Count



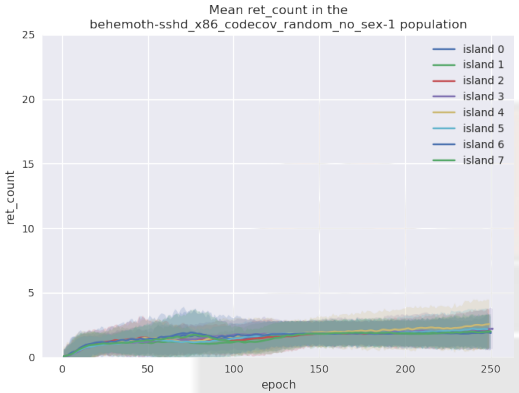


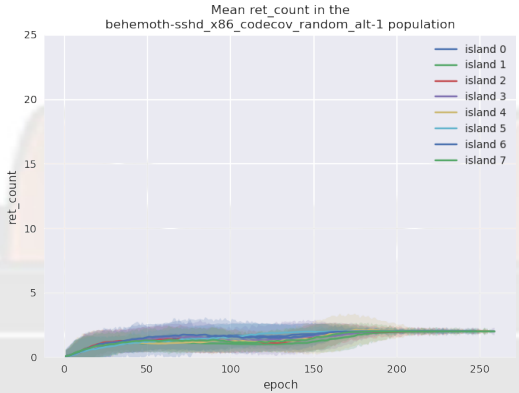


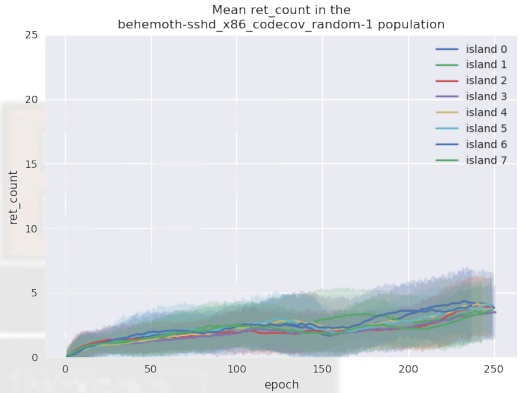


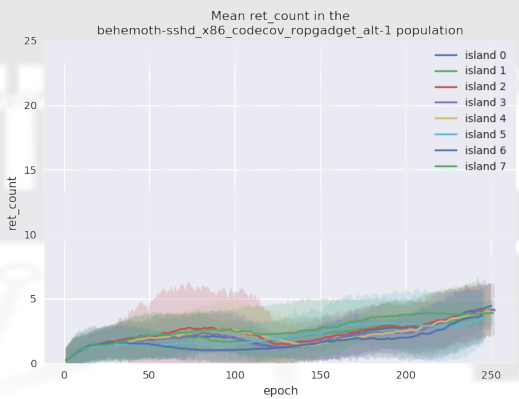


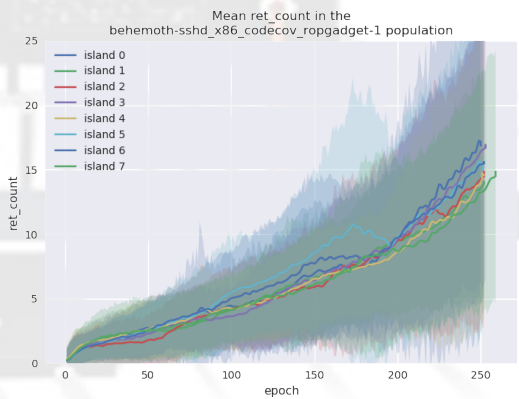
Return Count



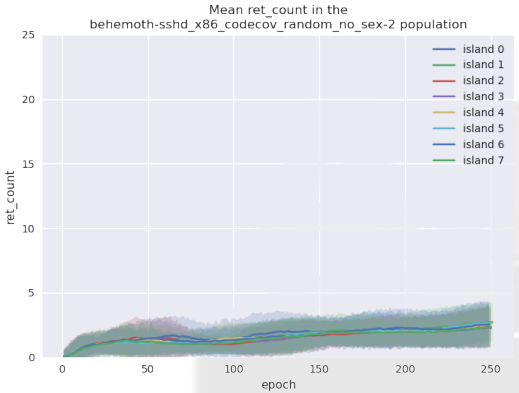


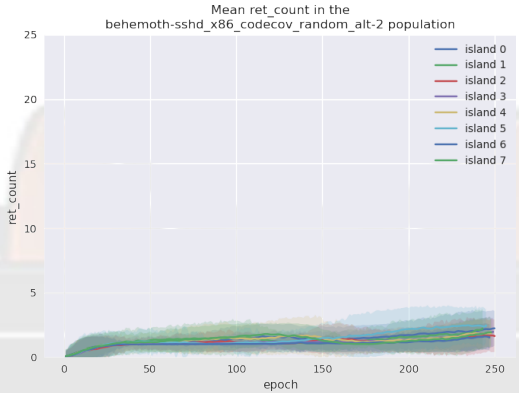


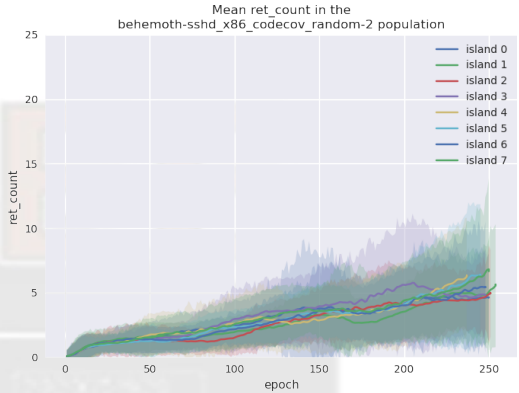


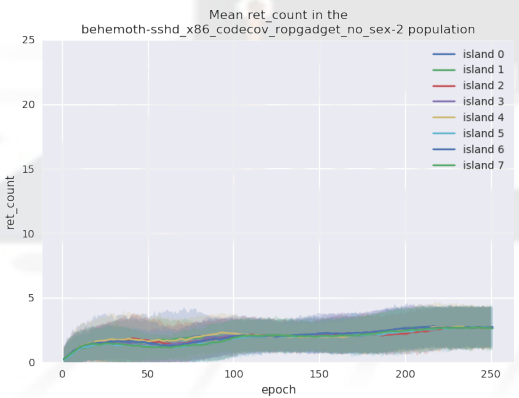


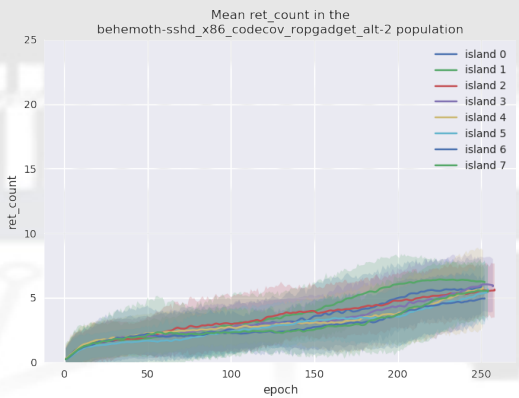
Return Count



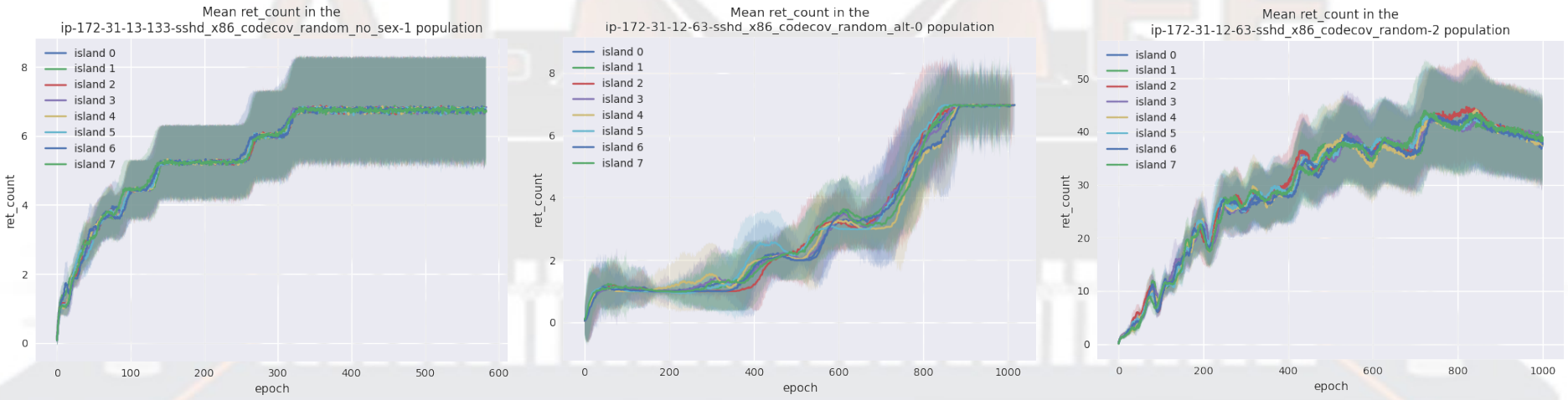




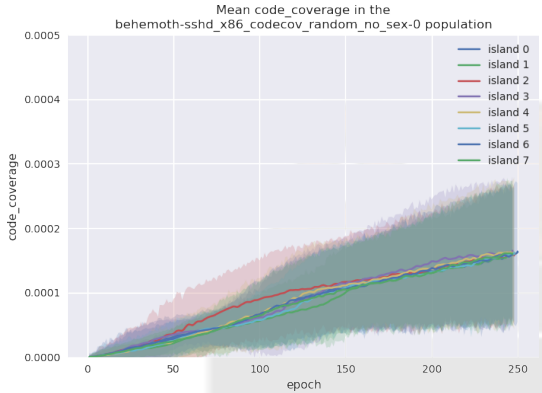


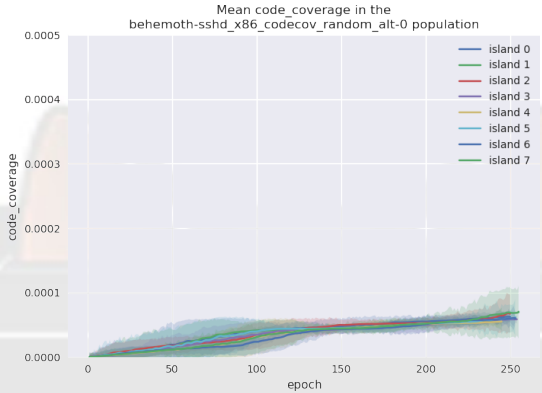


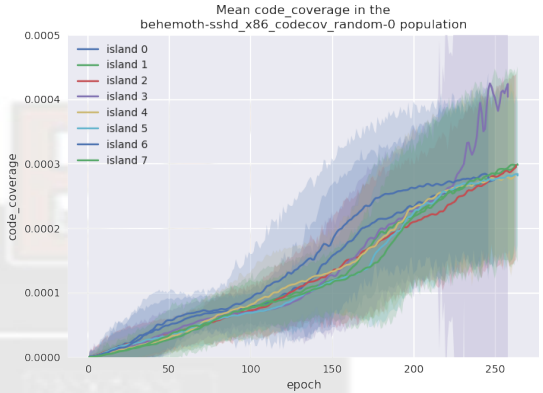
Return Count (long runs, not scaled)

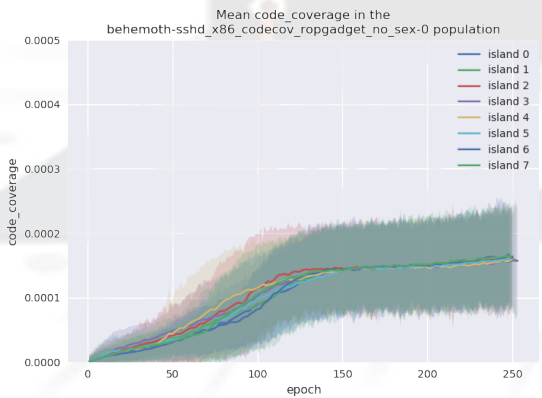


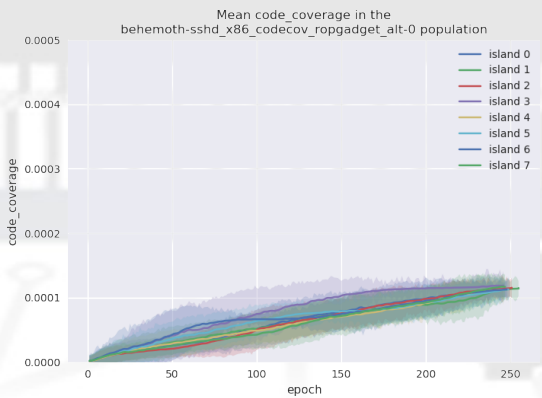
Code Coverage

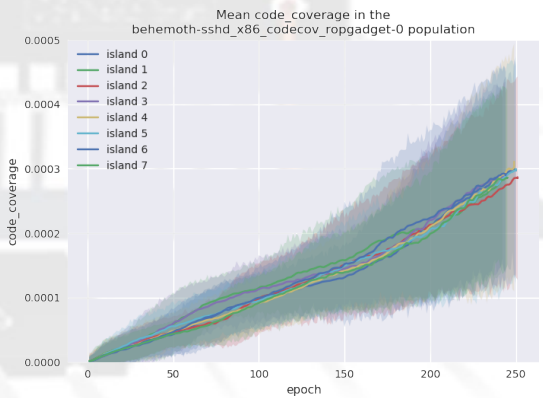




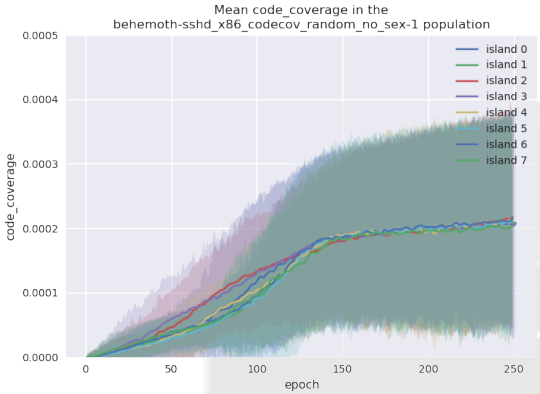


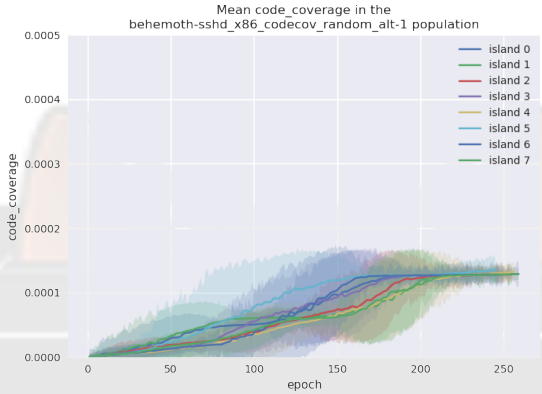


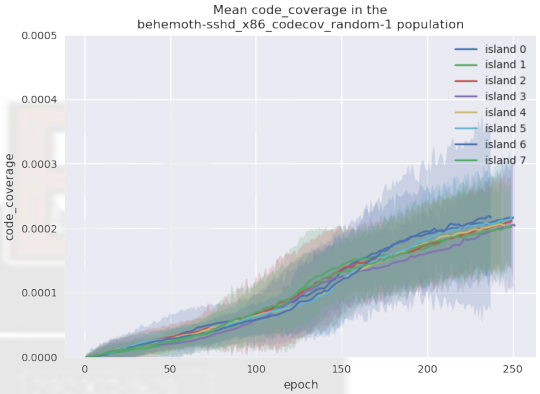


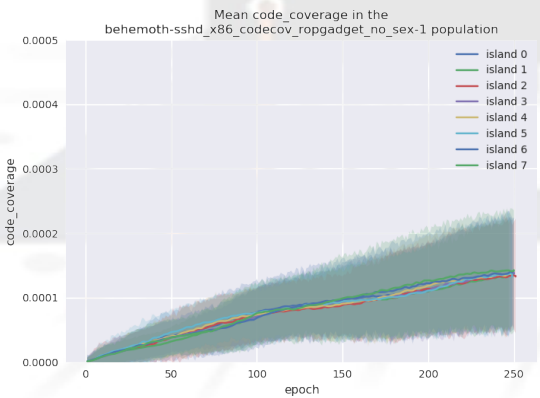


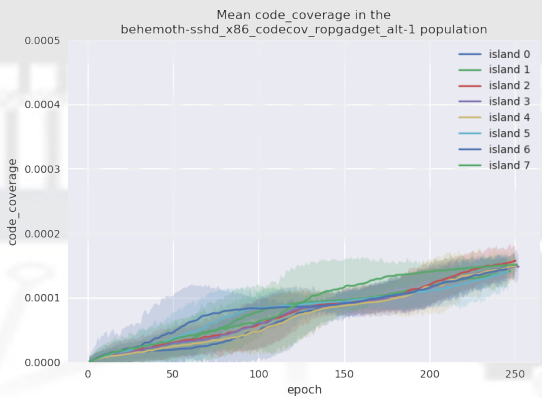
Code Coverage

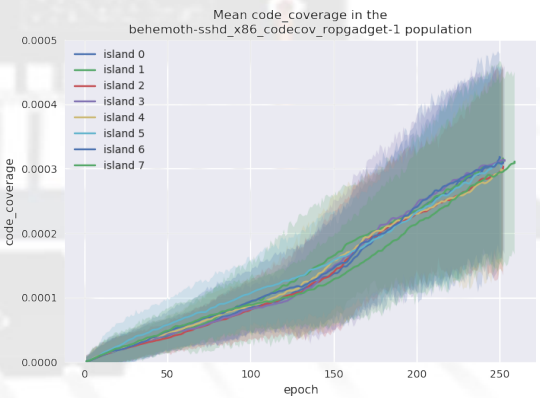




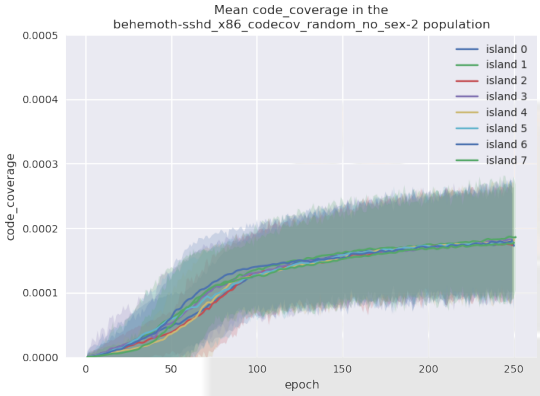


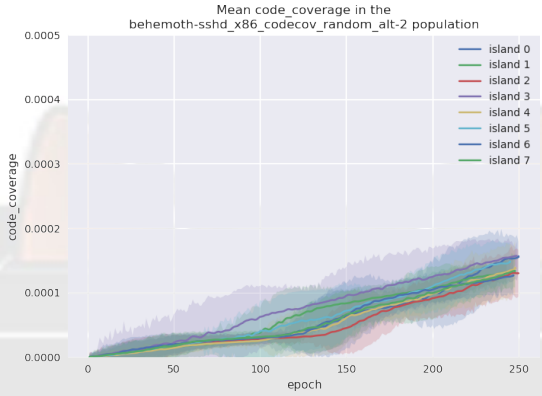


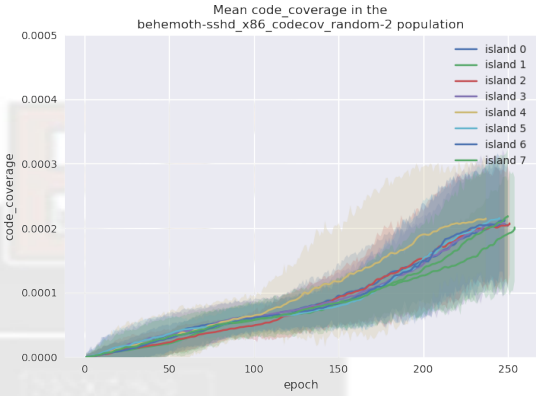


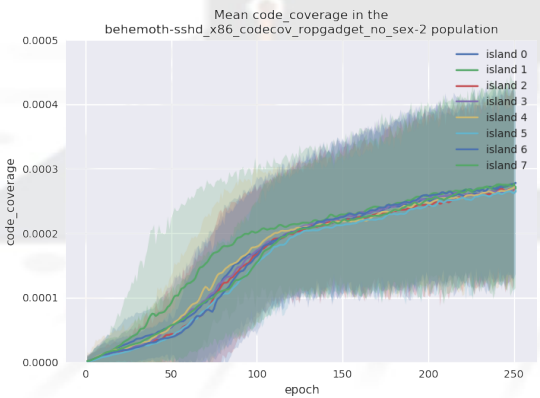


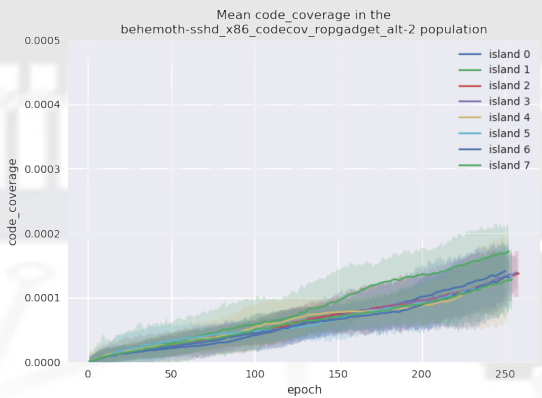
Code Coverage

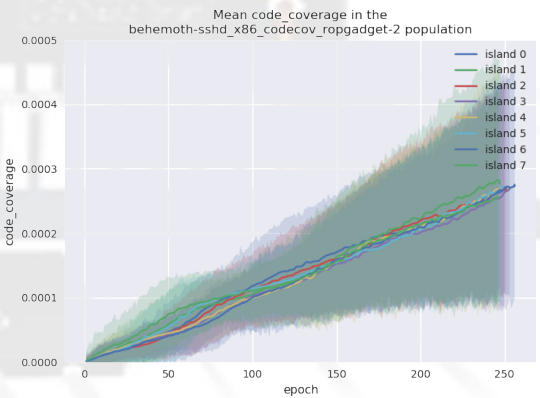




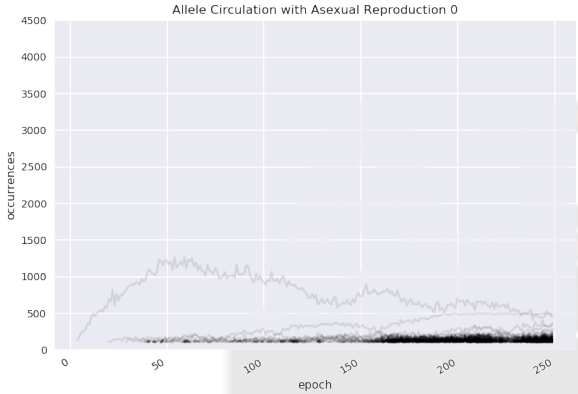


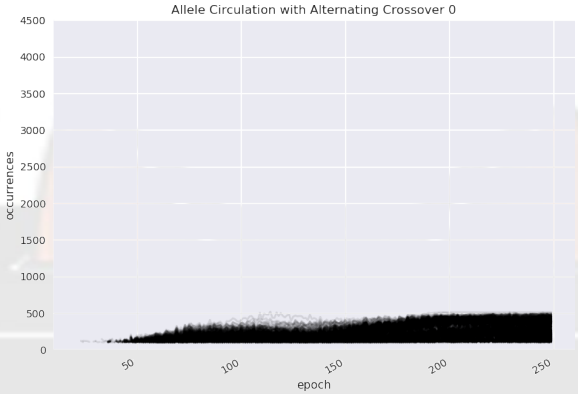


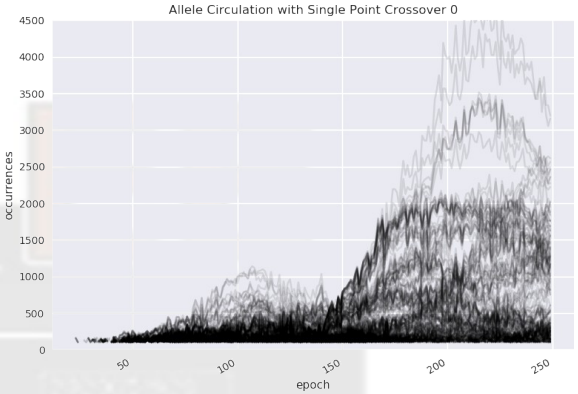


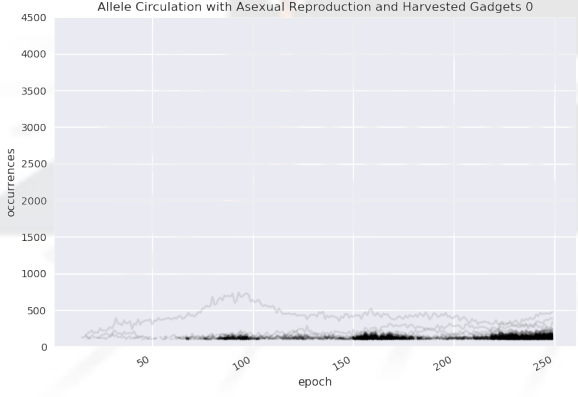


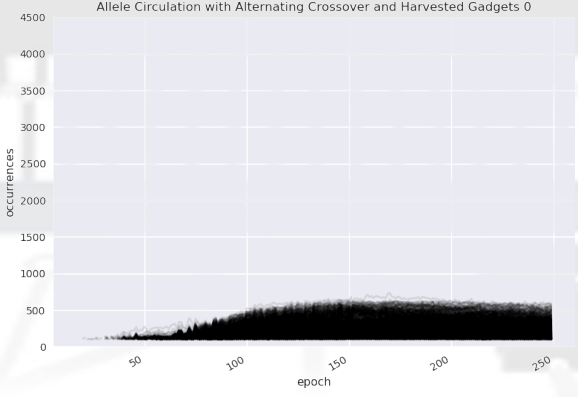
Allele Circulation

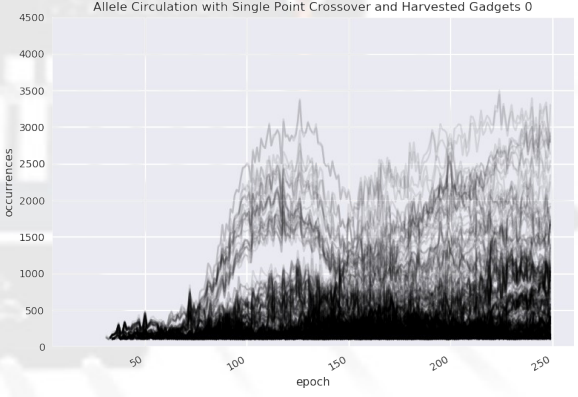




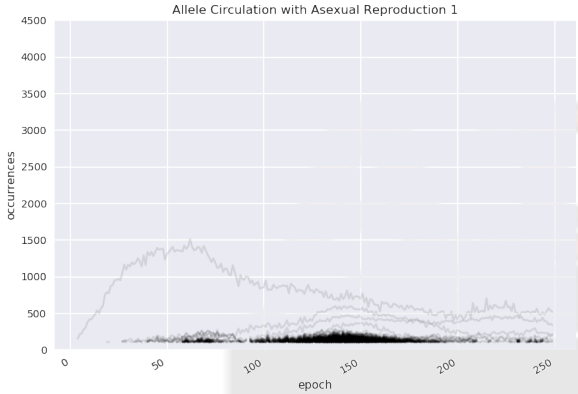


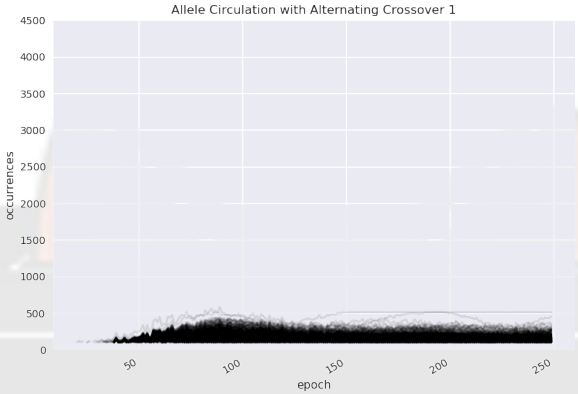


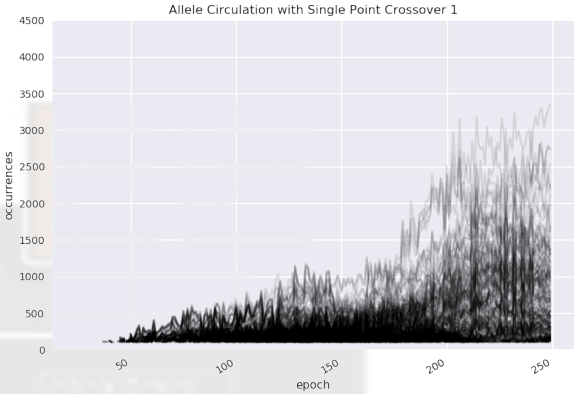


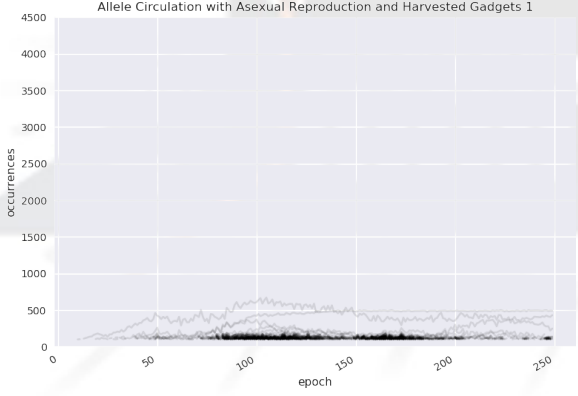


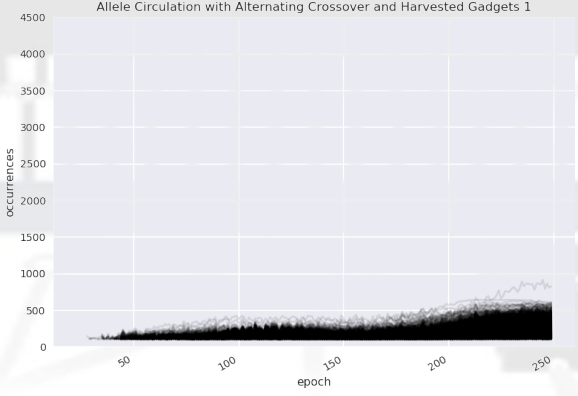
Allele Circulation

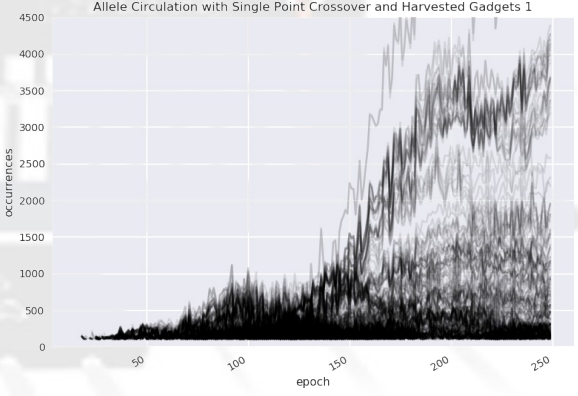




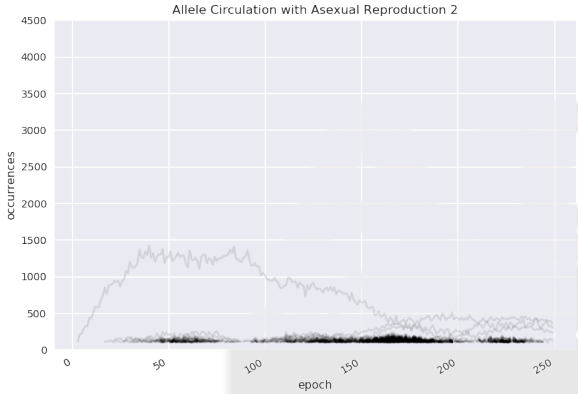


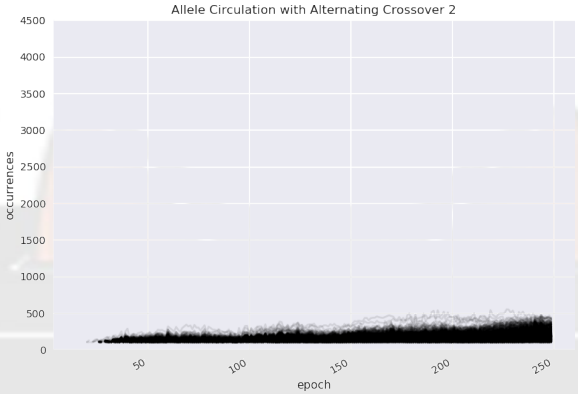


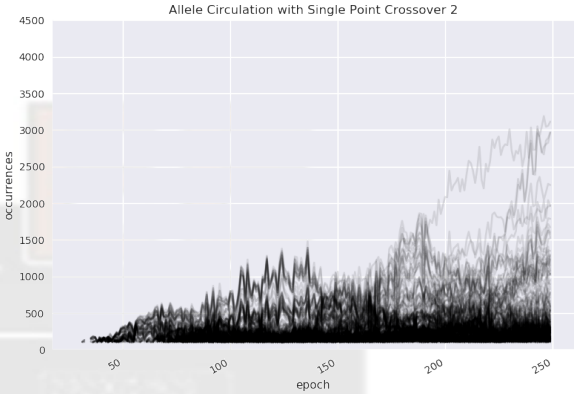


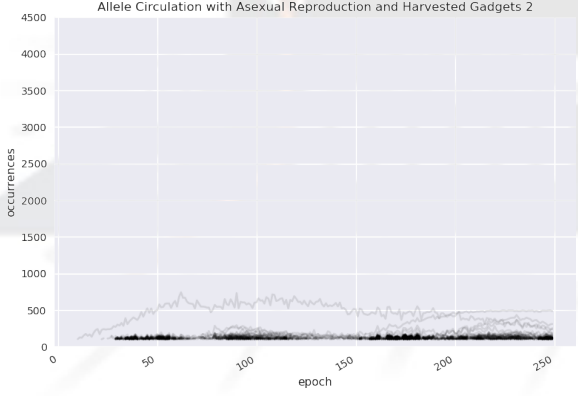


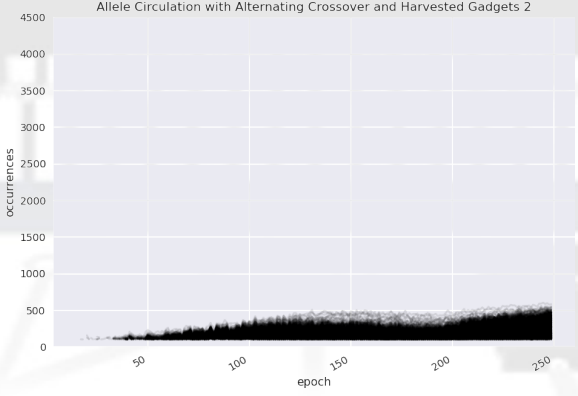
Allele Circulation

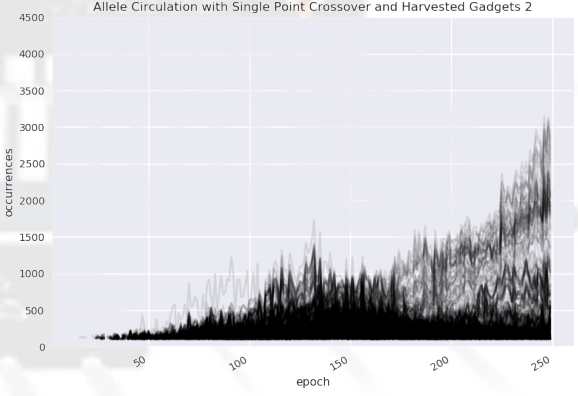


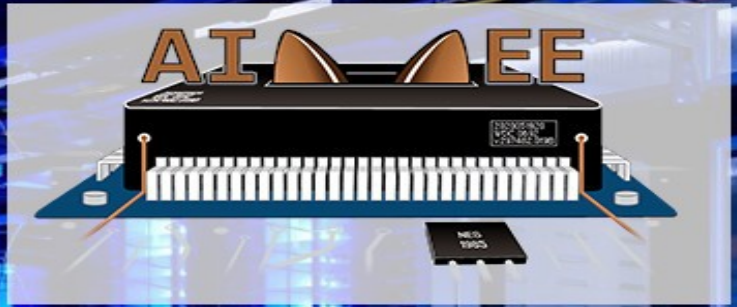






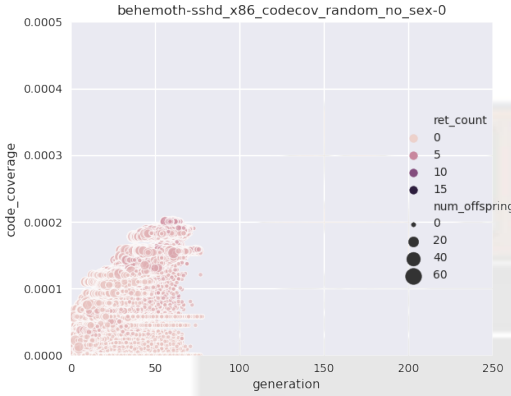


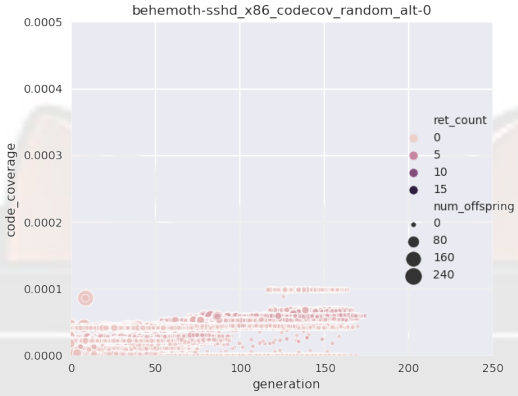


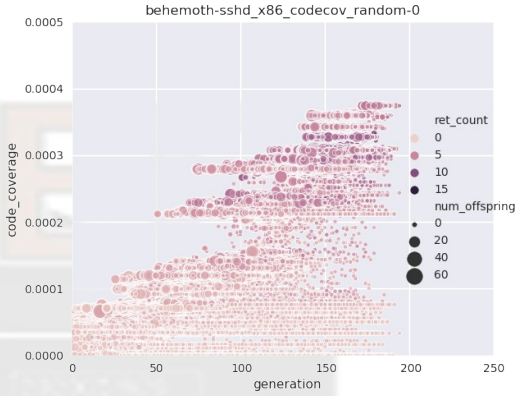


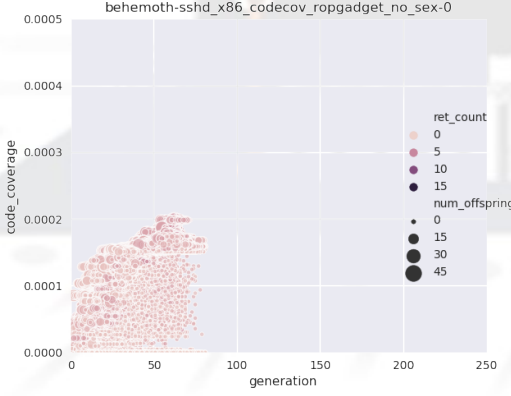
Generational distribution

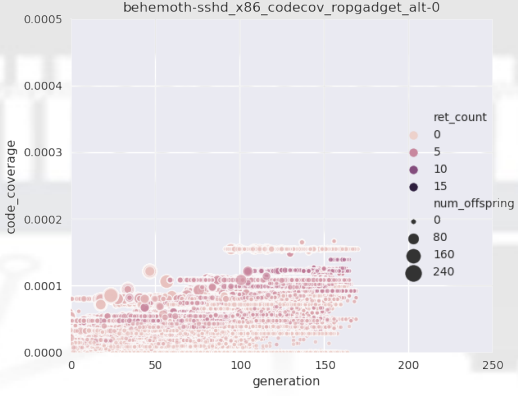
Generational Distribution

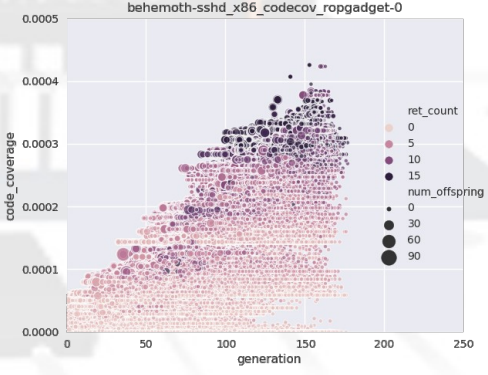




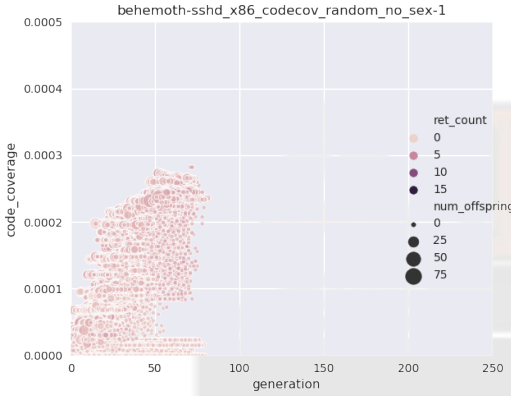


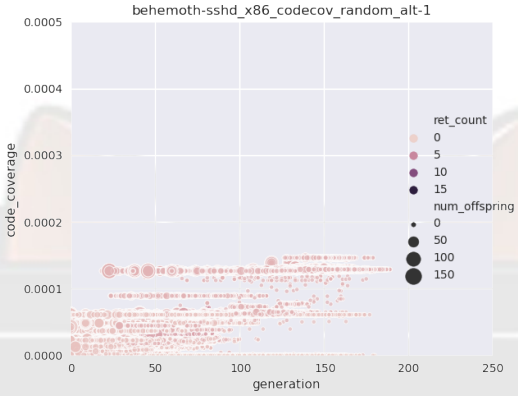


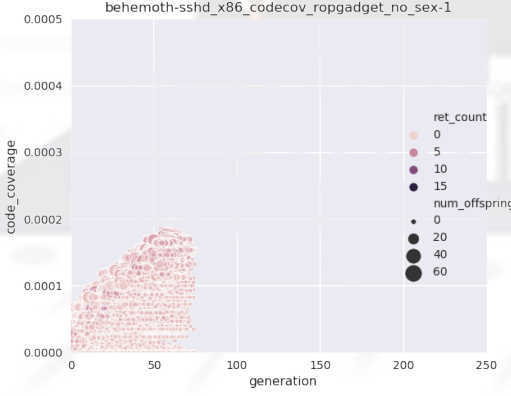


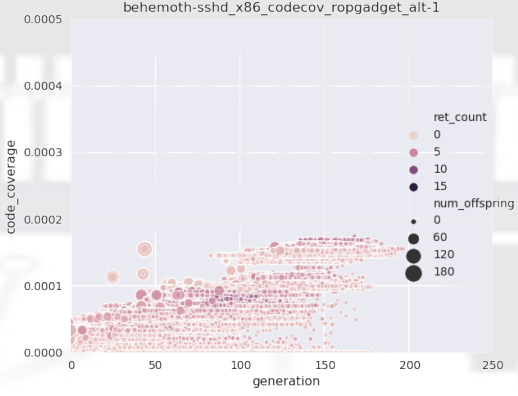


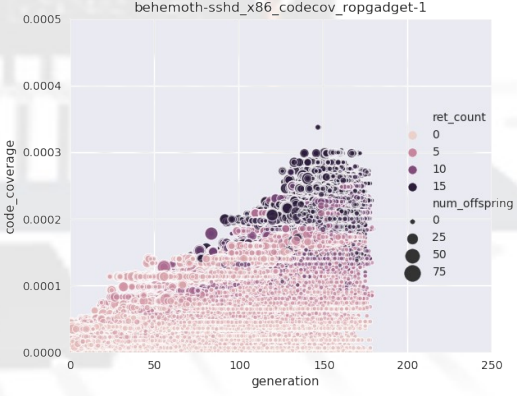
Generational Distribution



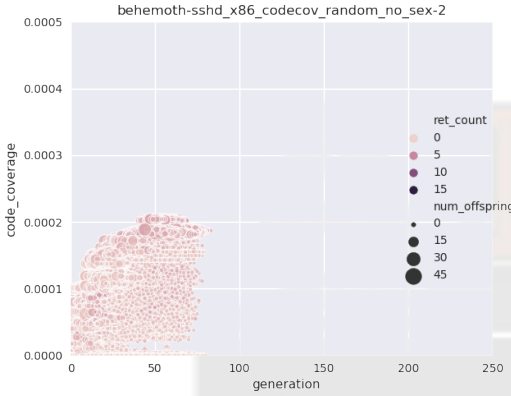


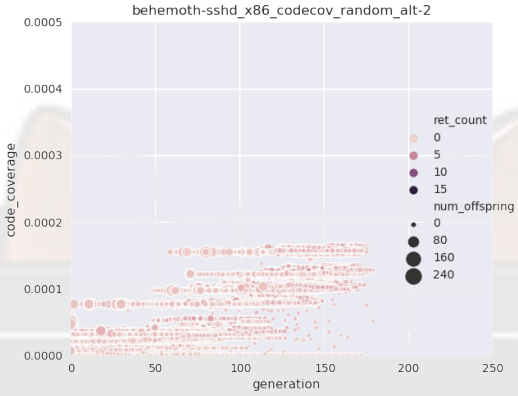


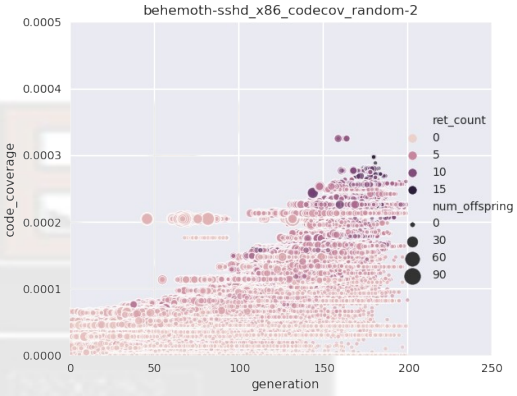


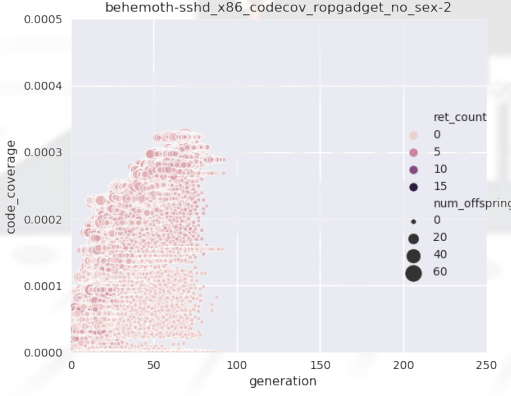


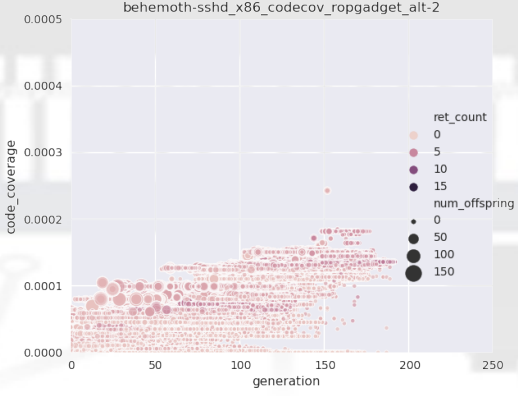
Generational Distribution

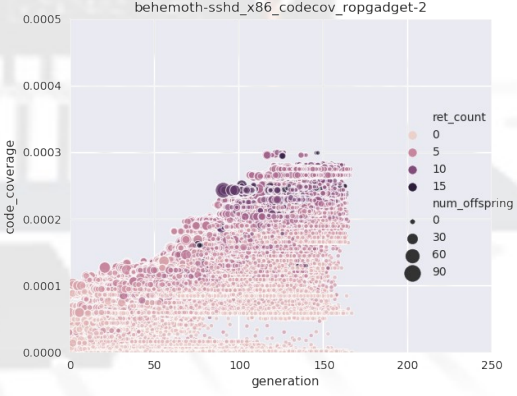












Experiment conclusions

Experiment conclusions

- Support for the *mixability theory of sexual reproduction?*
- Alternating crossover is not much better than asexual reproduction
- ROPgadget does not help asexual reproduction significantly
- Allele circulation is faster on single point crossover
- It is slowest with alternating crossover
- Higher return count correlates with code coverage

Register control

Register control

- Difficult to define proximity
  - Graph with state transitions: too expensive
  - Euclidean distance: too inaccurate
  - Hamming distance: too inaccurate
- How to handle indirections?
- How do we handle writable memory?

Register Control

- Weighted hamming distance
  - For each register and first m references:
    - Choose closest target, add 1 for each disagreement
    - Apply penalty if result is in different register
  - Sum all the results to get distance

Specimen #1

```
Name: wiles-flied-nooks-whipt, from island 0
Generation: 2736

Trace:
----
80b5dfa:        89 f0                                   mov eax, esi
80b5dfc:        8b 4c 24 54                             mov ecx, dword ptr [esp + 0x54]
80b5e00:        25 00 00 00 c0                          and eax, 0xc0000000
80b5e05:        01 c1                                   add ecx, eax
80b5e07:        03 44 24 58                             add eax, dword ptr [esp + 0x58]
80b5e0b:        81 e6 ff ff ff 3f                       and esi, 0x3fffffff
80b5e11:        89 c2                                   mov edx, eax
80b5e13:        74 39                                   je 0x80b5e4e
----
80b5e4e:        83 c4 3c                                add esp, 0x3c
80b5e51:        b8 01 00 00 00                          mov eax, 1
80b5e56:        5b                                      pop ebx
80b5e57:        5e                                      pop esi
80b5e58:        5f                                      pop edi
80b5e59:        5d                                      pop ebp
80b5e5a:        c3                                      ret
----
8075df7:        52                                      push edx
8075df8:        1c f6                                   sbb al, 0xf6
8075dfa:        c2 02 74                                ret 0x7402


Spidered register state:
EAX: 0xb
EBP: 0x81606d5 RX -> 0x312e2520 " %.1"
EBX: 0x81606a8 RX -> 0x6e69622f "/bin"
ECX: 0x8049633 RX -> 0x0
EDX: 0x0
EIP: 0x8075dfa RX -> 0xe7402c2
ESP: 0x8218150 RW (stack) -> 0x0
```

Specimen #2

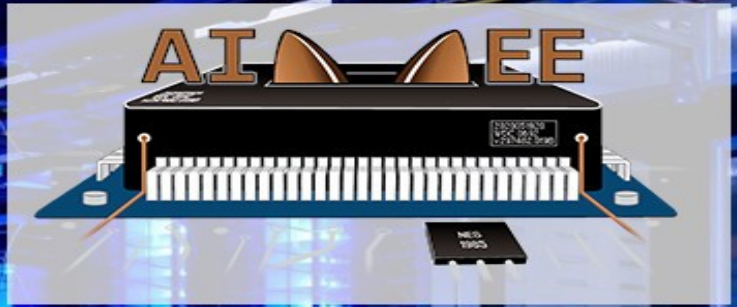
```
Name: corms-taxis-magma-wefts, from island 4
Generation: 951

Trace:
----
80badf1:        83 fb ff                cmp ebx, -1
80badf4:        74 0f                   je 0x80bae05
80badf6:        8d 4b 10                lea ecx, [ebx + 0x10]
80badf9:        31 d2                   xor edx, edx
80badfb:        39 4c 24 5c             cmp dword ptr [esp + 0x5c], ecx
80badff:        0f 85 99 01 00 00       jne 0x80baf9e
80baf9e:        83 c4 3c                add esp, 0x3c
80bafa1:        89 d0                   mov eax, edx
80bafa3:        5b                      pop ebx
80bafa4:        5e                      pop esi
80bafa5:        5f                      pop edi
80bafa6:        5d                      pop ebp
80bafa7:        c3                      ret
----
80badf1:        83 fb ff                cmp ebx, -1
80badf4:        74 0f                   je 0x80bae05
80badf6:        8d 4b 10                lea ecx, [ebx + 0x10]
80badf9:        31 d2                   xor edx, edx
80badfb:        39 4c 24 5c             cmp dword ptr [esp + 0x5c], ecx
80badff:        0f 85 99 01 00 00       jne 0x80baf9e
80baf9e:        83 c4 3c                add esp, 0x3c
80bafa1:        89 d0                   mov eax, edx
80bafa3:        5b                      pop ebx
80bafa4:        5e                      pop esi
80bafa5:        5f                      pop edi
80bafa6:        5d                      pop ebp
80bafa7:        c3                      ret
----
80badf1:        83 fb ff                cmp ebx, -1
80badf4:        74 0f                   je 0x80bae05
80badf6:        8d 4b 10                lea ecx, [ebx + 0x10]
80badf9:        31 d2                   xor edx, edx
80badfb:        39 4c 24 5c             cmp dword ptr [esp + 0x5c], ecx
80badff:        0f 85 99 01 00 00       jne 0x80baf9e
80baf9e:        83 c4 3c                add esp, 0x3c
80bafa1:        89 d0                   mov eax, edx
80bafa3:        5b                      pop ebx
80bafa4:        5e                      pop esi
80bafa5:        5f                      pop edi
80bafa6:        5d                      pop ebp
80bafa7:        c3                      ret
----
8088fa1:        7d 94                   jge 0x8088f37
8088f37:        ac                      lodsb al, byte ptr [esi]
8088f38:        c3                      ret
```

```
Spidered register state:
EAX: 0xb
EBP: 0x81e9182 RX -> 0xe00c0002
EBX: 0x8189e76 RX -> 0x6e69622f "/bin"
ECX: 0x80482dd RX -> 0x0
EDX: 0x0
EIP: 0x8088f38 RX -> 0xf13101c3
ESP: 0x82181f4 RW (stack) -> 0x80ad3ae RX -> 0x8b097400
```

Register Control

- Full solutions are rare (2 out of 10 trials with 1000 epochs)
- Required applying secondary novelty pressure
  - Count-min-sketch used to weight errors (so new errors are less important)
- Push VM did not prove better than bare ROP chains



Current status

ROPER 3: Slothrop

- Moving over to Julia
  - A robust scientific computing ecosystem
  - Interoperability with C(++) and Python
    - Writing Julia bindings for Unicorn was a pleasant experience
  - Makes for a good lingua franca, and facilitates collaboration
- Focusing on support for large populations, distributed over a network, rather than on single-host, efficiently threaded nodes

Next Steps: Problem

- Genetic programming is extremely effective, but uses a naive encoding of strings
- Strings are encoded as categorical data with distance/similarity a function of **string edit distance**
- “cat” → neighbors → “cot” - this doesn't reflect similarities in the meaning of the words
- With machine instructions, memory-adjacent machine instructions *can* have similar effects (or “meaning”) but this is far from guaranteed

Next Steps: Intuition

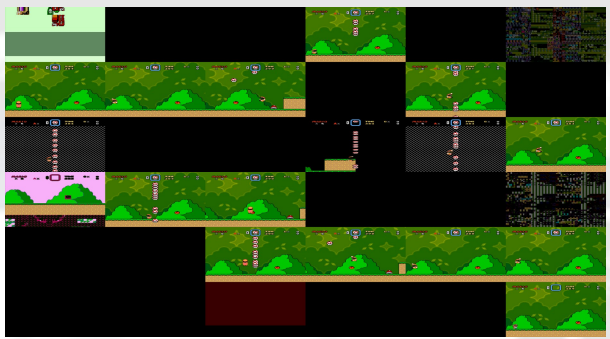
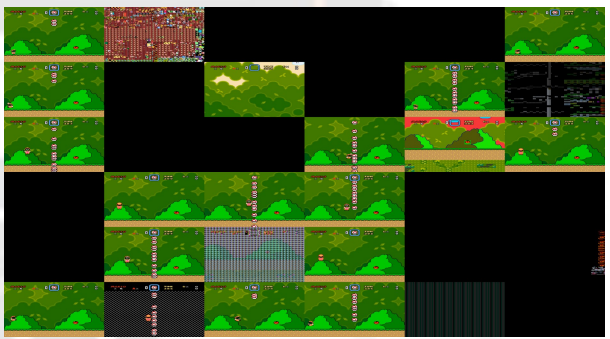
- Deep Learning already has an unreasonably effective solution to this problem: namely **Word2Vec**
- Word2Vec learns a vector representation of words that reflects their **contextual** similarity and difference
- This allows for **analogical reasoning** with extremely simple vector operations

Next Steps: Solution

- **Goal:** Reduce the amount of time for convergence in symbolic regression and increase transferability of solution
- Bootstrap an encoding by creating labeled data through the original genetic programming process, learn context
- Then switch to performing symbolic regression on the discovered latent space

Mario project

- Attempt at generalizing current results
- AI targets memory corruption glitches
- AI being used to automate exploitation (from induced weird states) in old videogames

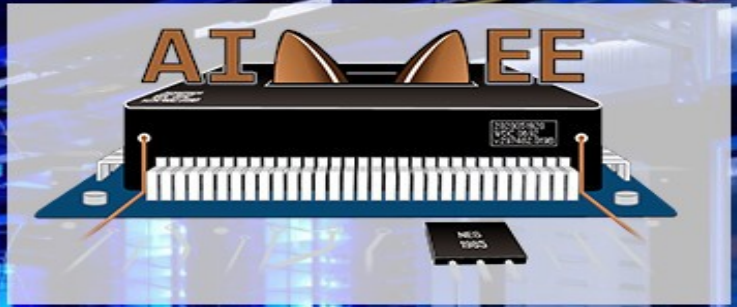




Schedule

- Jan 2021 Demonstrate generalized design against other systems (Mario project)
- March 2021 Final report
- June 2021 Academic papers submitted

Sources

- October report (and a lot of other information) on https://github.com/oblivia-simplex/berbalang/blob/master/org/october_report.org
- Berbalang source code: https://github.com/oblivia-simplex/berbalang
- Slothrop source code: https://github.com/oblivia-simplex/slothrop
- Thread-safe fork of Unicorn: https://github.com/oblivia-simplex/unicorn
- Julia Unicorn bindings: https://github.com/oblivia-simplex/unicorn-jl



Questions?

Lucca Fraser
lucca@special-circumstanc.es