InTELL
BY FOX IT

Evolving binary code with AGs

Who am I

Jesús Olmos

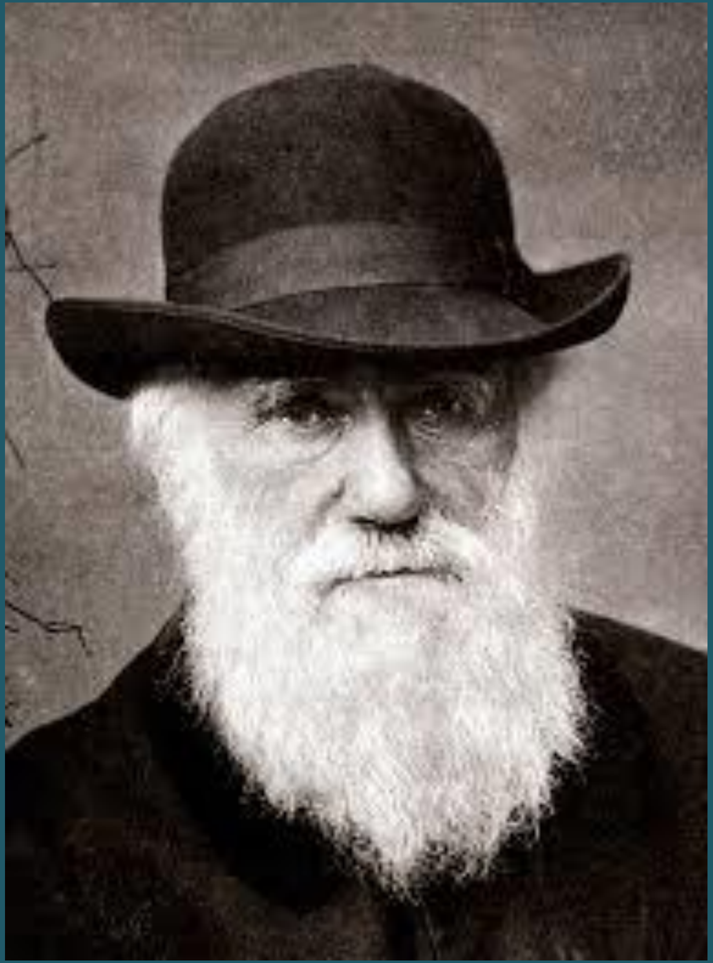
jolmos

@sha0coder

sha0@badchecksum.net
https://github.com/sha0coder

InTELL
BY FOX IT

Genetic Algorithms (GAs)



From evolving data to evolving code.

Automatic logic creation

Two aproaches:

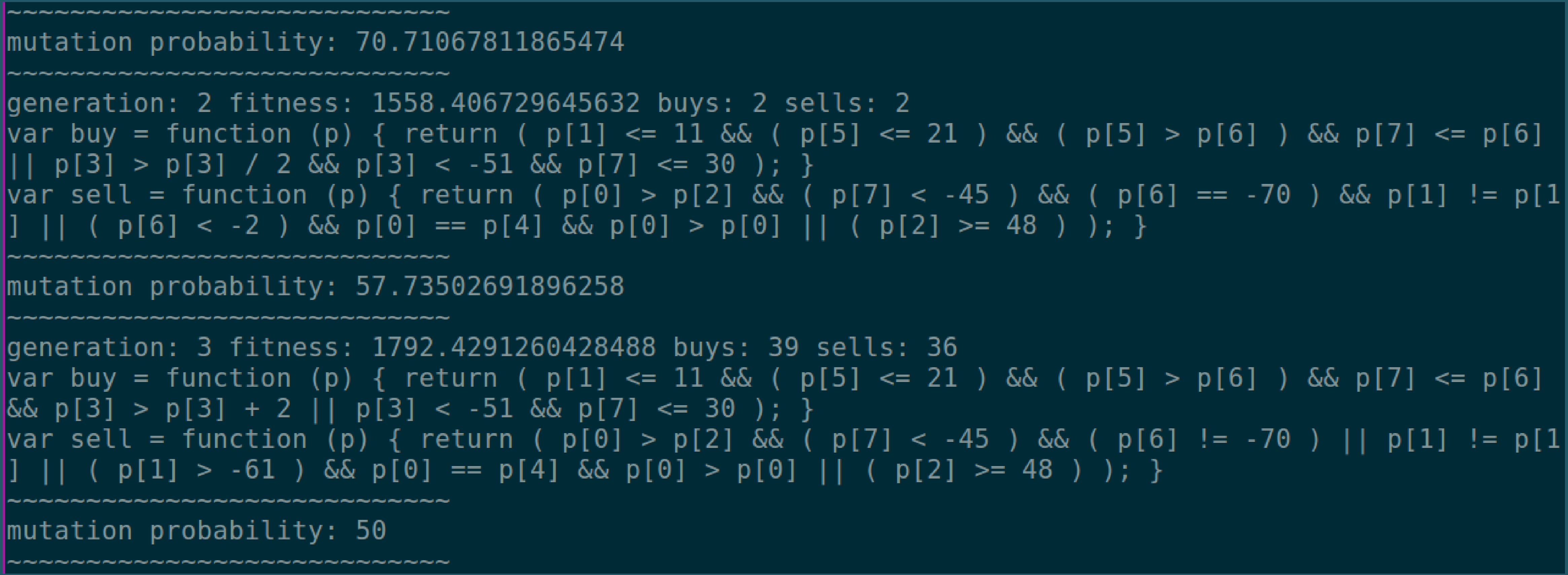
1. Providing all the details.
2. Providing a purpose. (test cases or evaluation function)

Measuring the evolution

Evolving trading strategy.

```
function buy(indicators) {

}

function sell(indicators) {

}
```

Measuring the evolution

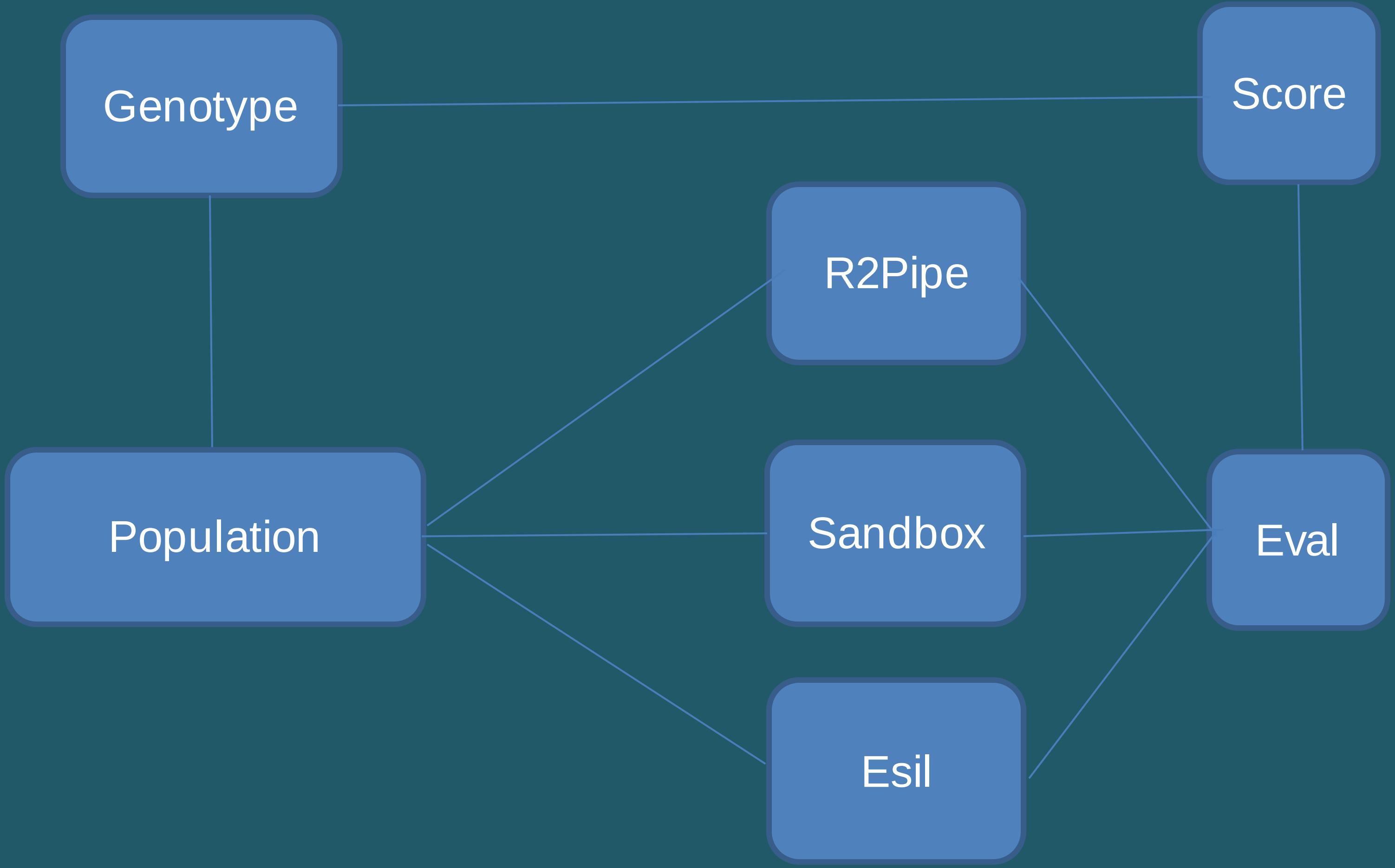


```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
mutation probability: 70.71067811865474
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
generation: 2 fitness: 1558.406729645632 buys: 2 sells: 2
var buy = function (p) { return ( p[1] <= 11 && ( p[5] <= 21 ) && ( p[5] > p[6] ) && p[7] <= p[6]
|| p[3] > p[3] / 2 && p[3] < -51 && p[7] <= 30 ); }
var sell = function (p) { return ( p[0] > p[2] && ( p[7] < -45 ) && ( p[6] == -70 ) && p[1] != p[1
] || ( p[6] < -2 ) && p[0] == p[4] && p[0] > p[0] || ( p[2] >= 48 ) ); }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
mutation probability: 57.73502691896258
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
generation: 3 fitness: 1792.4291260428488 buys: 39 sells: 36
var buy = function (p) { return ( p[1] <= 11 && ( p[5] <= 21 ) && ( p[5] > p[6] ) && p[7] <= p[6]
&& p[3] > p[3] + 2 || p[3] < -51 && p[7] <= 30 ); }
var sell = function (p) { return ( p[0] > p[2] && ( p[7] < -45 ) && ( p[6] != -70 ) || p[1] != p[1
] || ( p[1] > -61 ) && p[0] == p[4] && p[0] > p[0] || ( p[2] >= 48 ) ); }
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
mutation probability: 50
~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

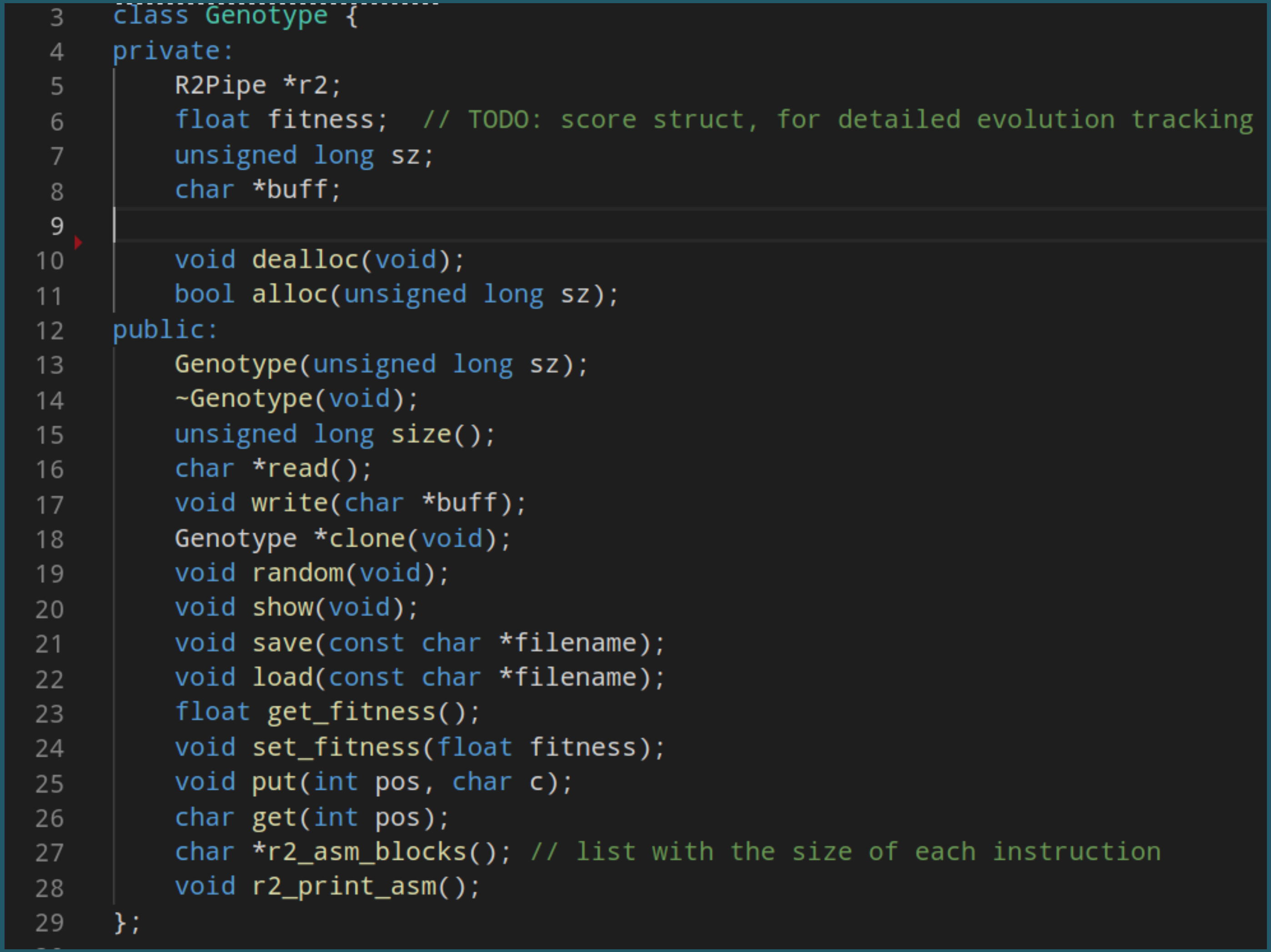
Let's evolve ASM code!!

Genetic Algorithm powered by ESIL

Design

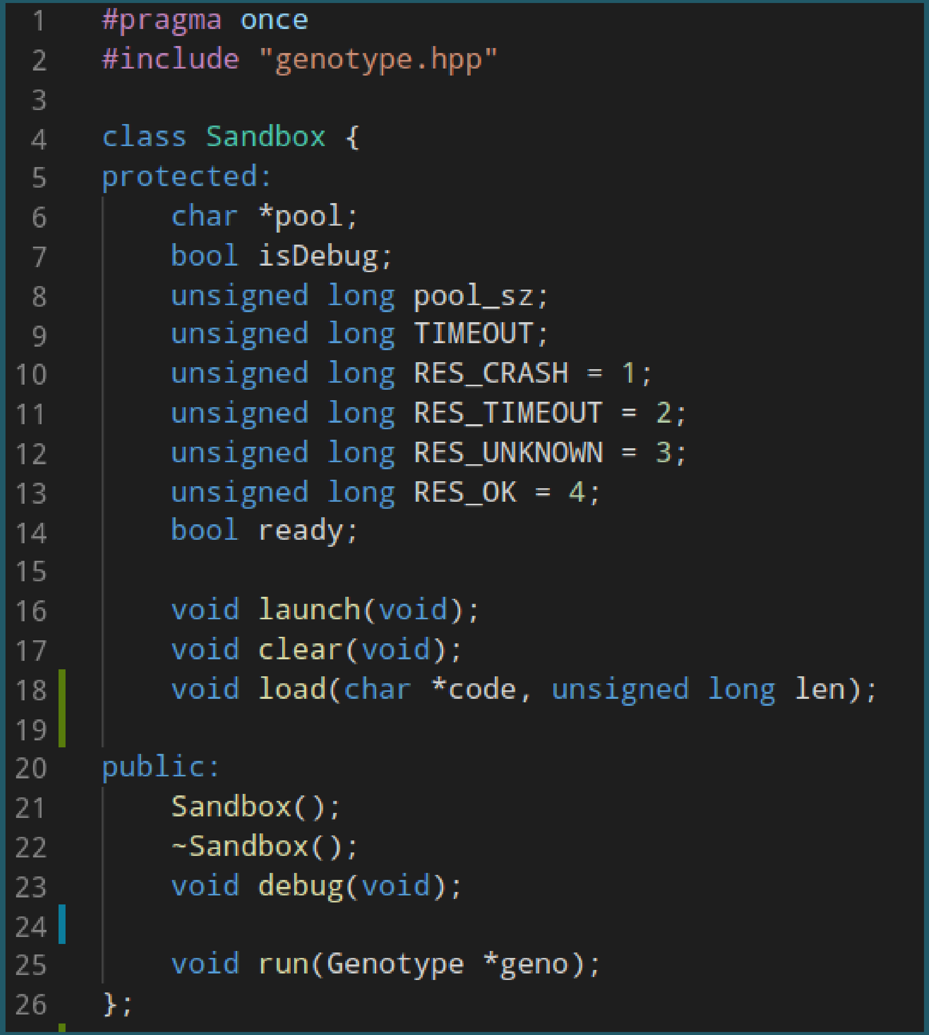


Genotype Definition



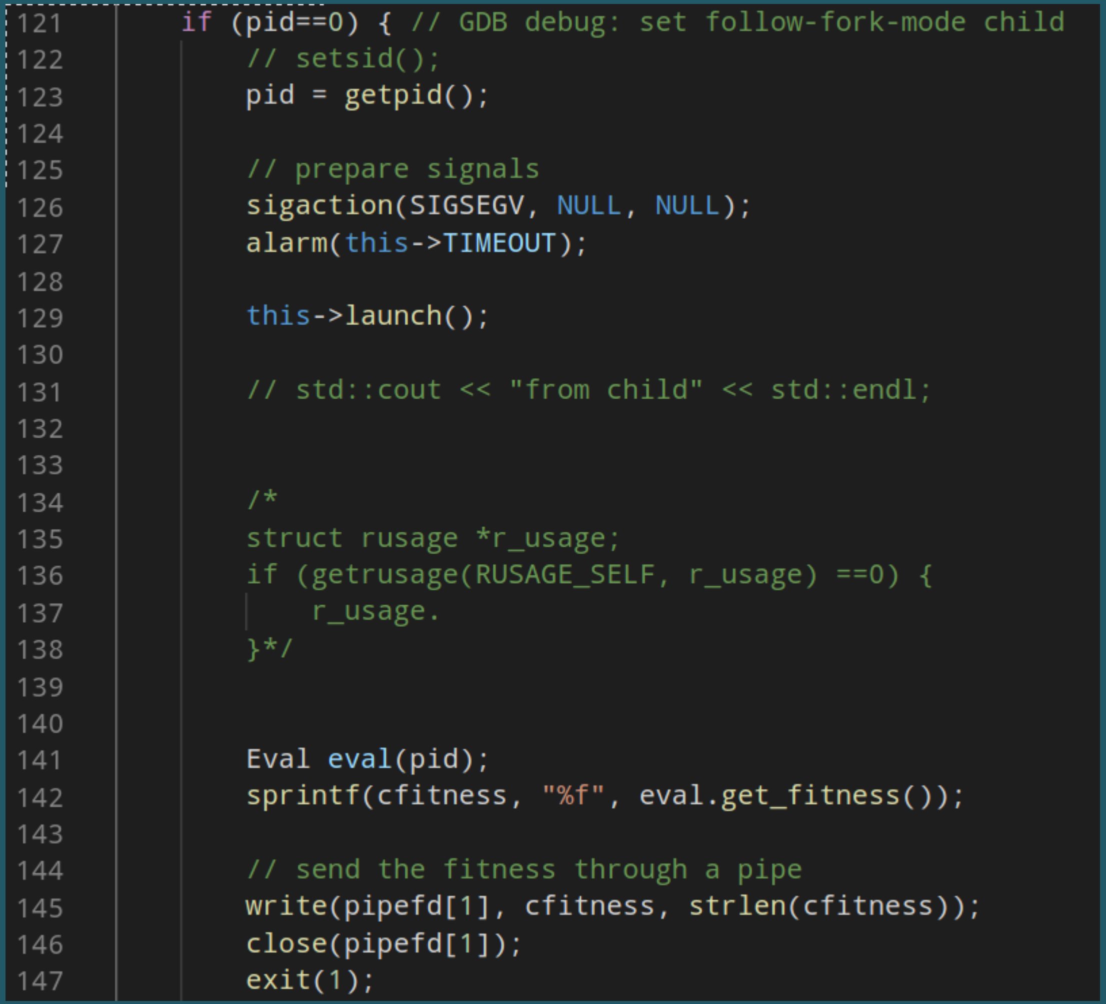
```
class Genotype {
private:
    R2Pipe *r2;
    float fitness;  // TODO: score struct, for detailed evolution tracking
    unsigned long sz;
    char *buff;

    void dealloc(void);
    bool alloc(unsigned long sz);
public:
    Genotype(unsigned long sz);
    ~Genotype(void);
    unsigned long size();
    char *read();
    void write(char *buff);
    Genotype *clone(void);
    void random(void);
    void show(void);
    void save(const char *filename);
    void load(const char *filename);
    float get_fitness();
    void set_fitness(float fitness);
    void put(int pos, char c);
    char get(int pos);
    char *r2_asm_blocks(); // list with the size of each instruction
    void r2_print_asm();
};
```

Sandbox definition



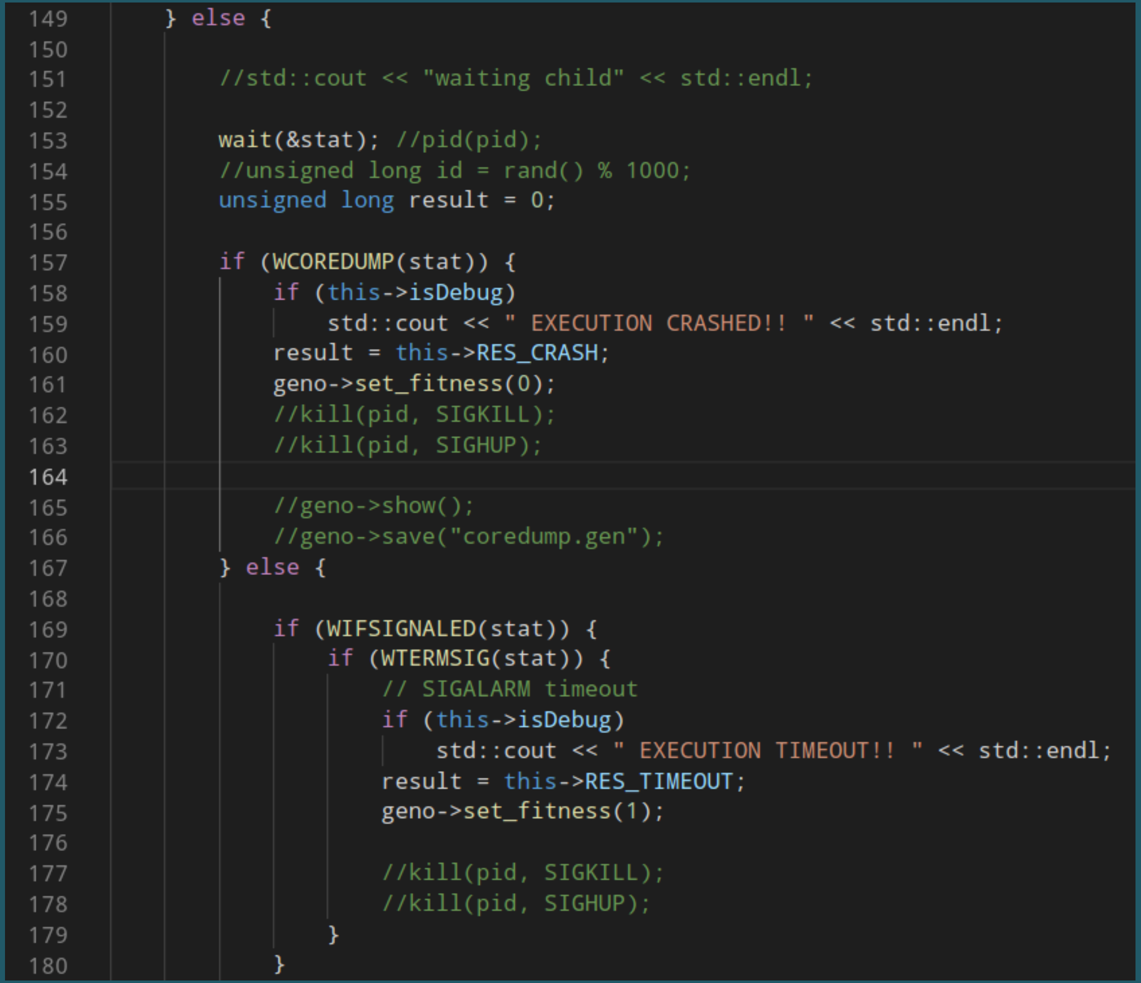
```
#pragma once
#include "genotype.hpp"

class Sandbox {
protected:
    char *pool;
    bool isDebug;
    unsigned long pool_sz;
    unsigned long TIMEOUT;
    unsigned long RES_CRASH = 1;
    unsigned long RES_TIMEOUT = 2;
    unsigned long RES_UNKNOWN = 3;
    unsigned long RES_OK = 4;
    bool ready;

    void launch(void);
    void clear(void);
    void load(char *code, unsigned long len);

public:
    Sandbox();
    ~Sandbox();
    void debug(void);

    void run(Genotype *geno);
};
```

Sandbox child process



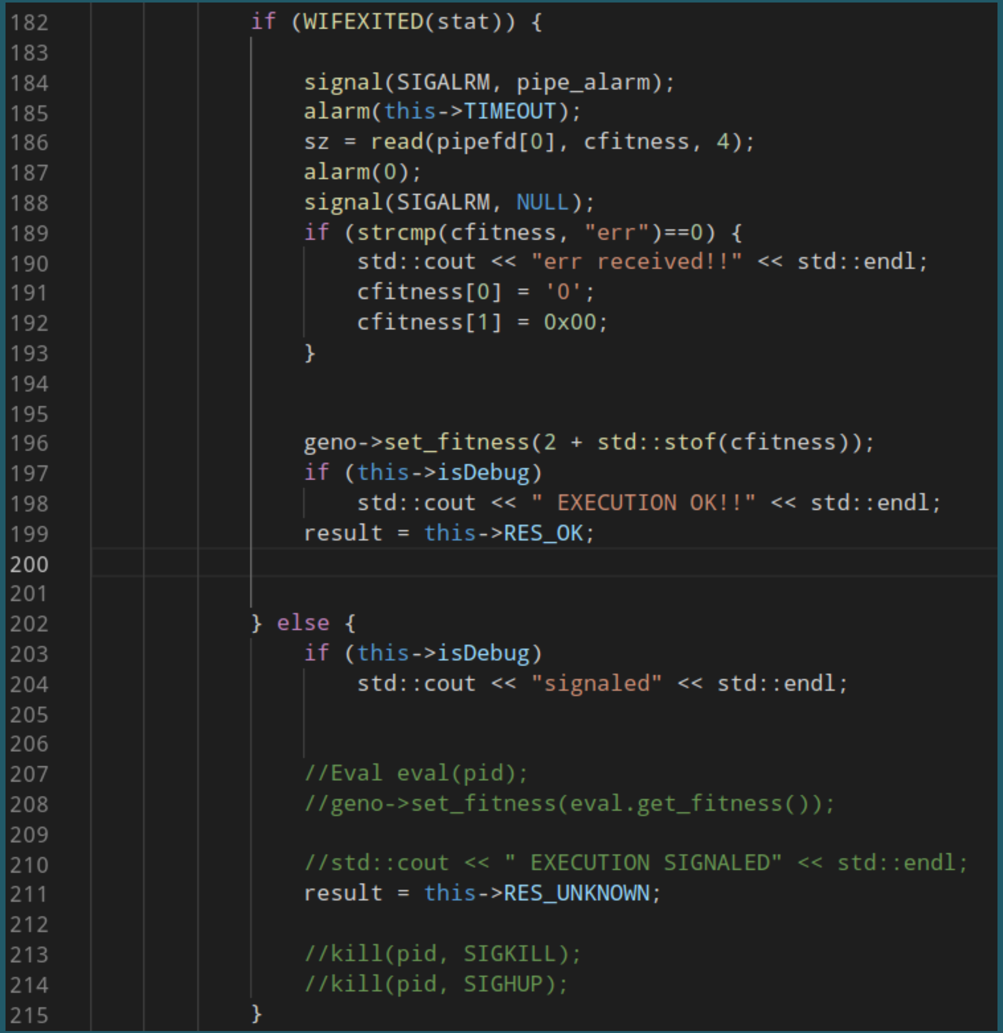
```
    if (pid==0) { // GDB debug: set follow-fork-mode child
        // setsid();
        pid = getpid();

        // prepare signals
        sigaction(SIGSEGV, NULL, NULL);
        alarm(this->TIMEOUT);

        this->launch();

        // std::cout << "from child" << std::endl;


        /*
        struct rusage *r_usage;
        if (getrusage(RUSAGE_SELF, r_usage) ==0) {
            r_usage.
        }*/


        Eval eval(pid);
        sprintf(cfitness, "%f", eval.get_fitness());

        // send the fitness through a pipe
        write(pipefd[1], cfitness, strlen(cfitness));
        close(pipefd[1]);
        exit(1);
```

Sandbox parent process



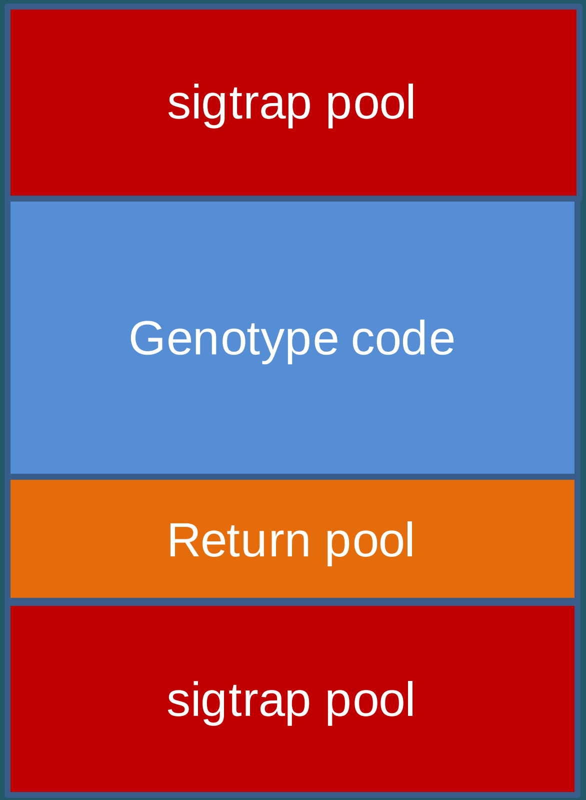
```
    } else {

        //std::cout << "waiting child" << std::endl;

        wait(&stat); //pid(pid);
        //unsigned long id = rand() % 1000;
        unsigned long result = 0;

        if (WCOREDUMP(stat)) {
            if (this->isDebug)
                std::cout << " EXECUTION CRASHED!! " << std::endl;
            result = this->RES_CRASH;
            geno->set_fitness(0);
            //kill(pid, SIGKILL);
            //kill(pid, SIGHUP);

            //geno->show();
            //geno->save("coredump.gen");
        } else {

            if (WIFSIGNALED(stat)) {
                if (WTERMSIG(stat)) {
                    // SIGALARM timeout
                    if (this->isDebug)
                        std::cout << " EXECUTION TIMEOUT!! " << std::endl;
                    result = this->RES_TIMEOUT;
                    geno->set_fitness(1);

                    //kill(pid, SIGKILL);
                    //kill(pid, SIGHUP);
                }
            }
```

Sandbox parent process



```
            if (WIFEXITED(stat)) {

                signal(SIGALRM, pipe_alarm);
                alarm(this->TIMEOUT);
                sz = read(pipefd[0], cfitness, 4);
                alarm(0);
                signal(SIGALRM, NULL);
                if (strcmp(cfitness, "err")==0) {
                    std::cout << "err received!!" << std::endl;
                    cfitness[0] = '0';
                    cfitness[1] = 0x00;
                }


                geno->set_fitness(2 + std::stof(cfitness));
                if (this->isDebug)
                    std::cout << " EXECUTION OK!!" << std::endl;
                result = this->RES_OK;


            } else {
                if (this->isDebug)
                    std::cout << "signaled" << std::endl;


                //Eval eval(pid);
                //geno->set_fitness(eval.get_fitness());

                //std::cout << " EXECUTION SIGNALED" << std::endl;
                result = this->RES_UNKNOWN;

                //kill(pid, SIGKILL);
                //kill(pid, SIGHUP);
            }
```

Sandbox



sigtrap pool

Genotype code

Return pool

sigtrap pool

Mutation

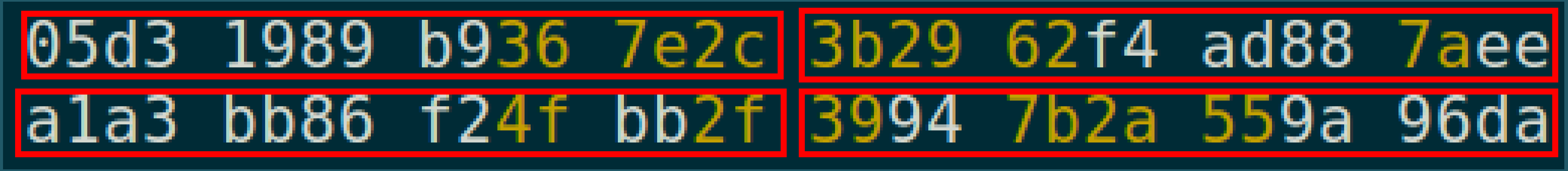
- GAs must use very low mutation probability.
- Hight mutation = random search --> Don’t converge.

- Different probabilities: opcode, operads, inmediates.

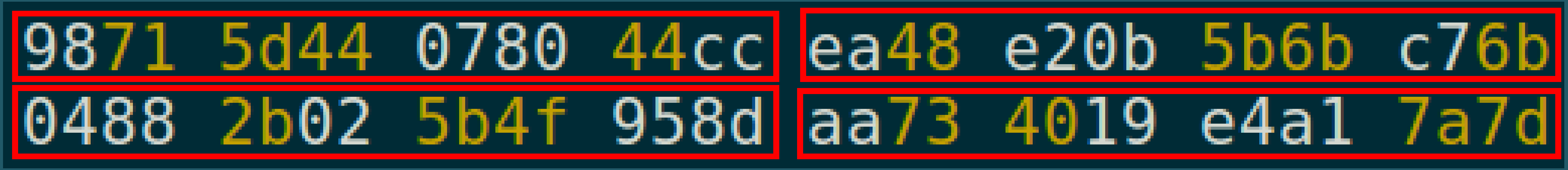
- From exploration to optimization.

Crossover

Byte level crossover:



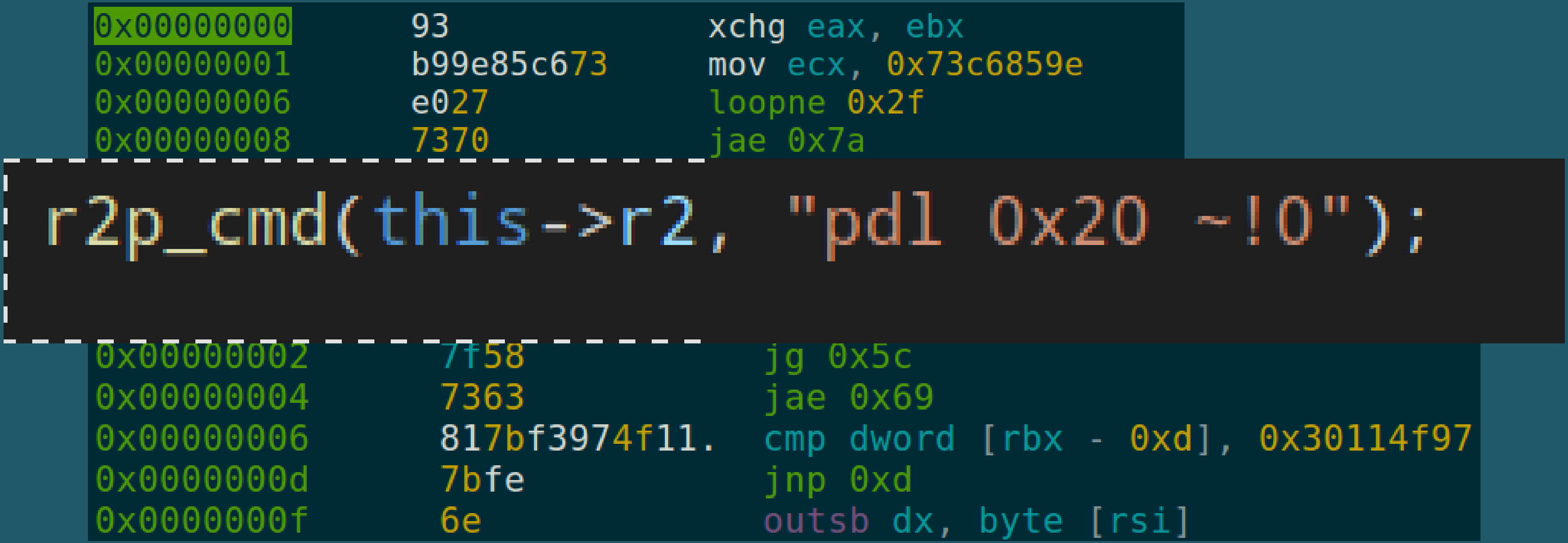
```
05d3 1989 b936 7e2c 3b29 62f4 ad88 7aee
a1a3 bb86 f24f bb2f 3994 7b2a 559a 96da
```



```
9871 5d44 0780 44cc ea48 e20b 5b6b c76b
0488 2b02 5b4f 958d aa73 4019 e4a1 7a7d
```

Crossover

Instruction level crossover: (also basic-block level crosover)



```
0x00000000        93                xchg eax, ebx
0x00000001        b99e85c673        mov ecx, 0x73c6859e
0x00000006        e027              loopne 0x2f
0x00000008        7370              jae 0x7a
```

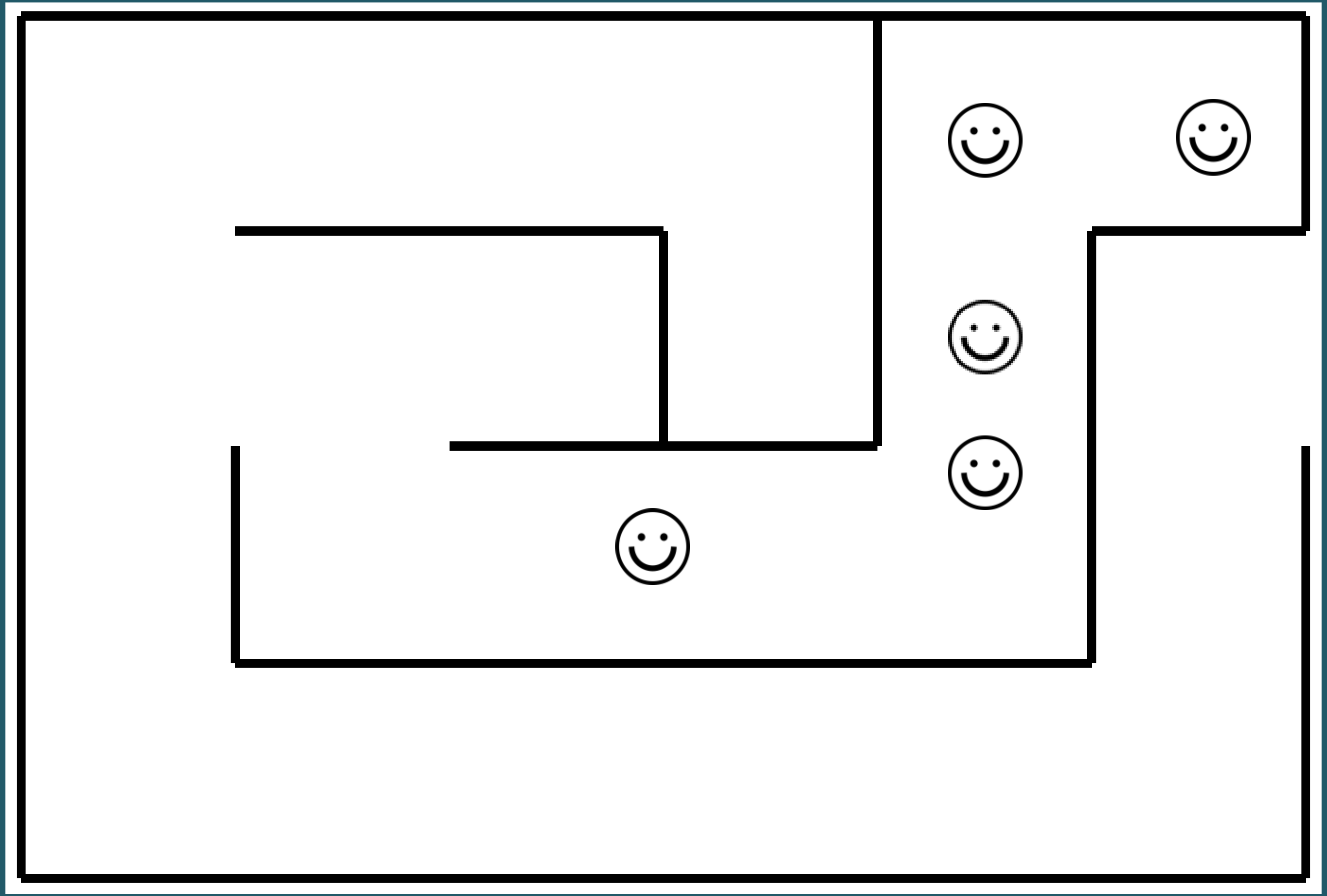
```
r2p_cmd(this->r2, "pdl 0x20 ~!0");
```

```
0x00000002        7f58              jg 0x5c
0x00000004        7363              jae 0x69
0x00000006        817bf3974f11.     cmp dword [rbx - 0xd], 0x30114f97
0x0000000d        7bfe              jnp 0xd
0x0000000f        6e                outsb dx, byte [rsi]
```

Evaluation

Error = distance

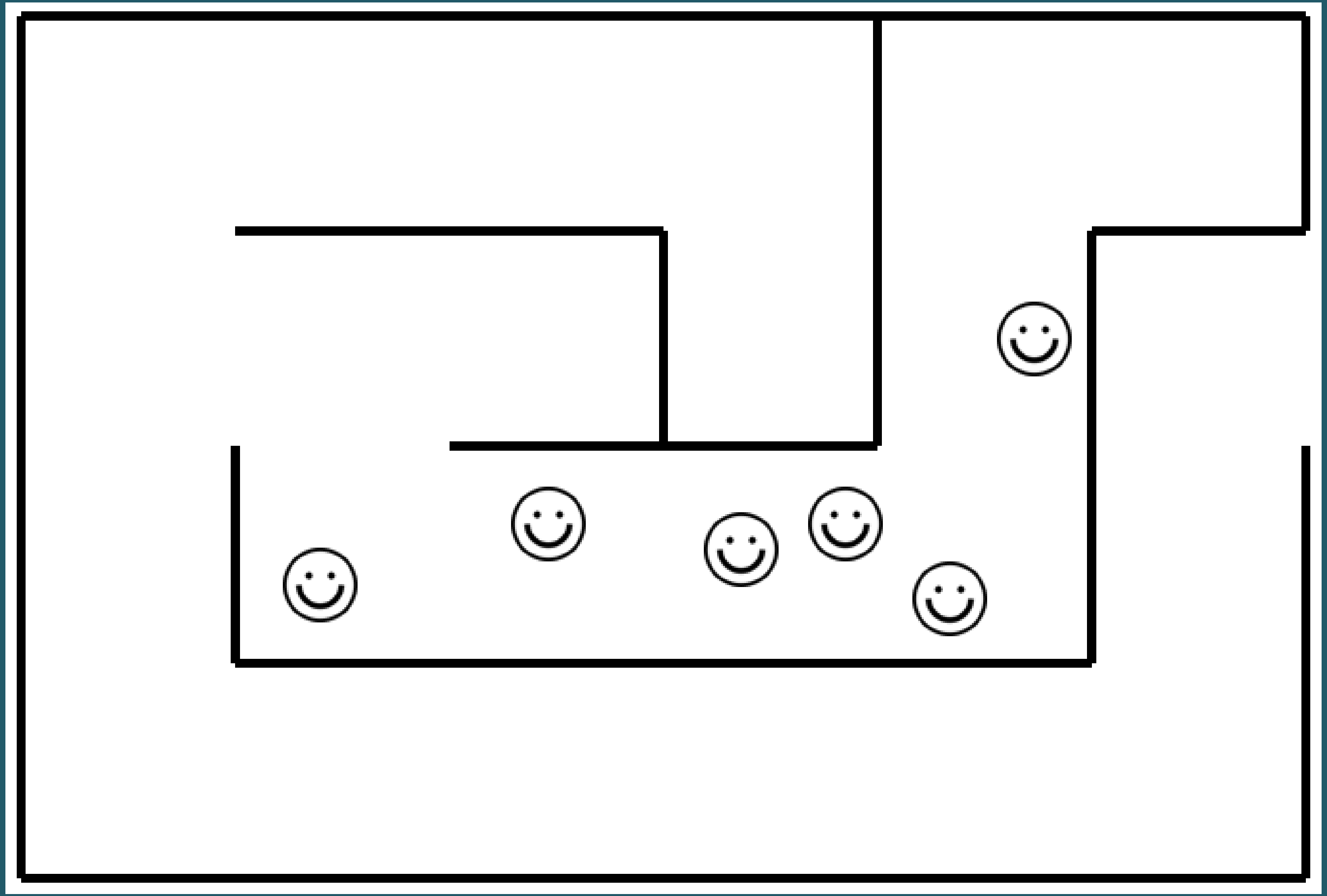
(exploration vs optimization)



Evaluation

Error = distance

(exploration vs optimization)



Evaluation

Dynamic evaluation.

- Must end the execution properly:
    - No segfaults, traps, etc ...
    - In less than TIMEOUT seconds.

- Measure process constants (oom, load, cpu, ram, ...)
- Fork + sandboxing vs emulation + esil

Evaluation

Static evaluation (r2pipe)

Proper sytax:

- only valid instructions
- no ret instructions

Radare branch prediction:

- avoid out of scope branches

Sys-calls increase the bonus

Validation

Dataset based problems requires a validation.

Use Cases?

- Triggering vulnerabilities? (requires a complex EF)
- The perfect r2wars warrior.
- Local DoS.
- Remote DoS.

- Search problems.
- Optimization problems.

Demo time

Genotype size: 0x20

Mutation probability: (8/g)%

Crossover rate: P*0.05

Github resources

Rust test-case solver:
https://github.com/sha0coder/spocky

C++ evolving assembly code:
https://github.com/sha0coder/predator

InTELL
BY FOX IT

Thanks.