

# Programação Genética + JIT

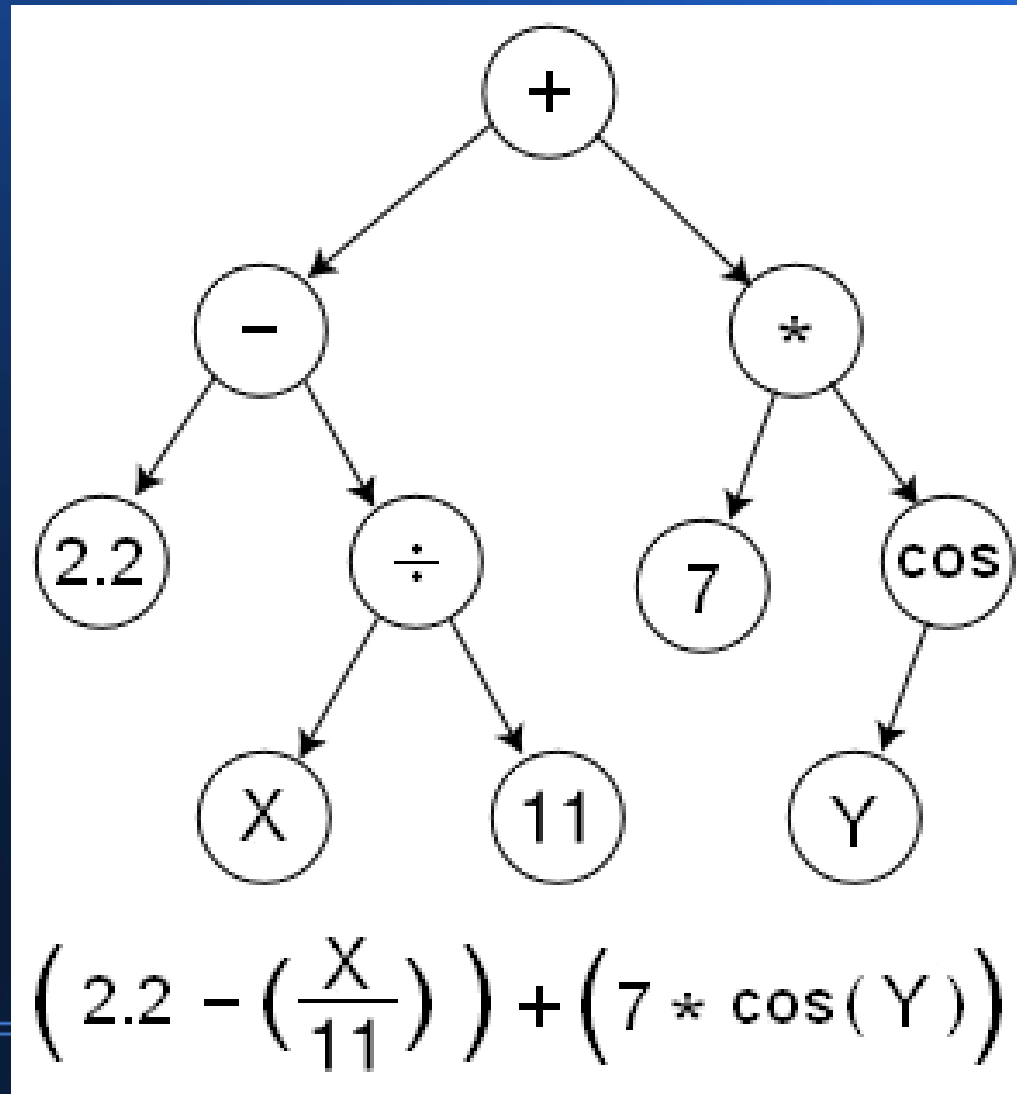
Tiago Camolesi Flora

Nº USP 5655201

# Programação Genética + JIT

- Programação Genética
  - Mesma premissa de computação evolucionária
  - População é composta de "programas"
  - Em geral, os programas são organizados em árvores e seguem um paradigma funcional
  - Possuem maior capacidade de generalização do que outros métodos evolutivos

# Programação Genética + JIT



# Programação Genética + JIT

- Projeto desenvolvido:
  - Implementação simplificada\* de um sistema de programação genética, usando um compilador just in time de expressões

\*Sem funções ou procedures

# Programação Genética + JIT

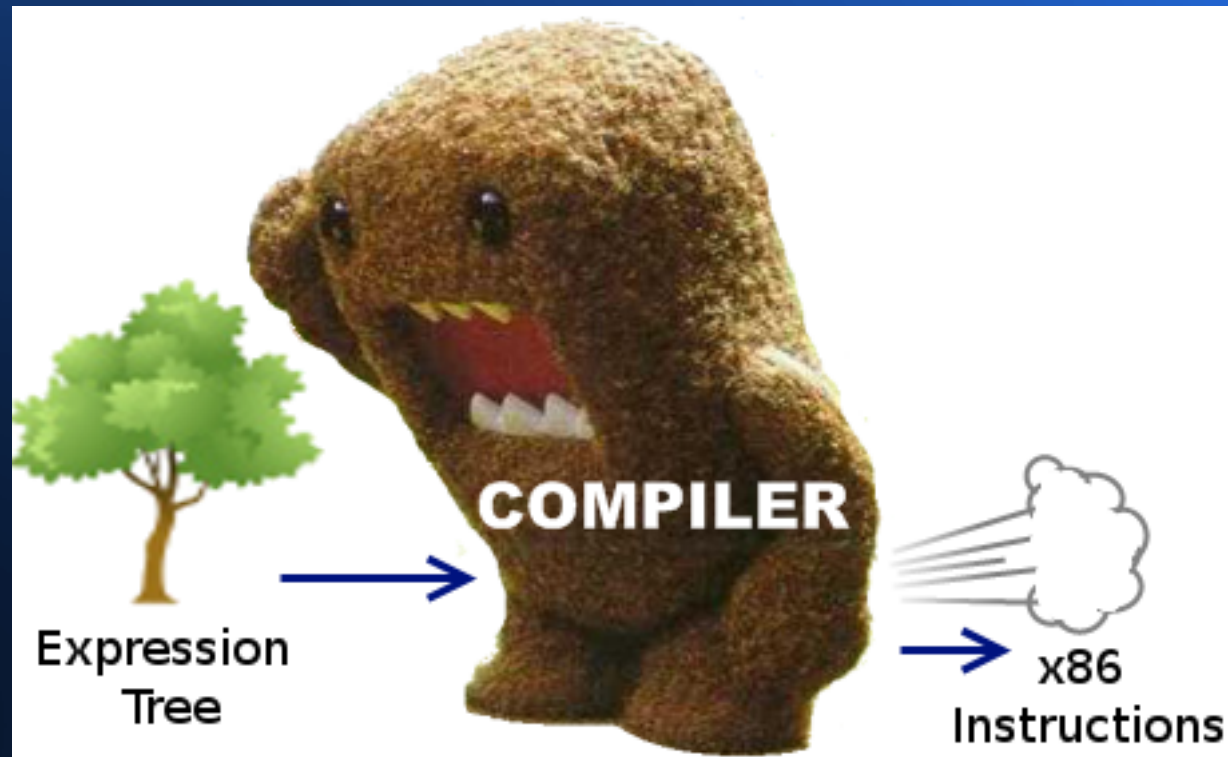
- Projeto desenvolvido:
  - Motivação:
    - Interpretação de expressões em árvores é um processo lento
    - Expressões devem ser interpretadas centenas de vezes para o cálculo de fitness dos indivíduos da população

# Programação Genética + JIT

- Overview
  - Um objeto é responsável por transformar uma árvore de expressões em uma função diretamente executável.

# Programação Genética + JIT

- Overview



# Programação Genética + JIT

- Overview

- O código deve ser armazenado numa página executável de memória.
- Pode-se obter este tipo de memória através do `mmap()` no Linux, ou `VirtualAlloc()` no Windows.
- Problema: Menor tamanho de página com permissões específicas: 4KB
- Desperdício de espaço e pobre localidade de cache



# Programação Genética + JIT

- Overview

- Solução: utilizar um objeto como pool de recursos de memória
- `AssemblerFactory`: responsável por criar os objetos `Assembler` e gerenciar a memória por eles utilizada.
- Dono de um grande espaço contíguo de memória executável e o divide entre os assemblers.

# Programação Genética + JIT

- Exemplo de utilização:

```
24 void __testAssemblerC() {  
25     RandomElementParams par;  
26     getRandomElementParams(&par);  
27     par.variablesRange[0] = 0;  
28     par.variablesRange[1] = 1;  
29     par.constantRange[0] = -15;  
30     par.constantRange[1] = 15;  
31     setRandomElementParams(&par);  
32     sin[((( $0 / cos[(((8.59 * -8.54) + $0)]) * 8.93)])]  
34     AssemblerFactory factory = createAssemblerFactory(512,  
35     Assembler a = createAssembler(factory);  
36     ETTree t = genRandomTree(7);  
37     beginAssembly(a);  
38     assembleTree(a, t);  
39     finishAssembly(a);  
40  
41     JITFunction fn = getJitFunction(a);  
42     float x[] = { 5 };  
43     float f = fn(x);  
44 }
```

End of assembler dump.  
(gdb) disassemble 0x003be200 0x003be240  
Dump of assembler code from 0x3be200 to 0x3be240:  
0x003be200: push ebx  
0x003be201: push edi  
0x003be202: mov ebx,0x3be3ff  
0x003be207: mov edi,DWORD PTR [esp+0xc]  
0x003be20b: fld DWORD PTR [edi+0x0]  
0x003be20e: fld DWORD PTR [ebx-0x4]  
0x003be211: fld DWORD PTR [ebx-0x8]  
0x003be214: fmulp st(1),st  
0x003be216: fld DWORD PTR [edi+0x0]  
0x003be219: faddp st(1),st  
0x003be21b: fcos  
0x003be21d: fdivrp st(1),st  
0x003be21f: fld DWORD PTR [ebx-0xc]  
0x003be222: fmulp st(1),st  
0x003be224: fsin  
0x003be226: pop edi  
0x003be227: pop ebx  
0x003be228: ret  
0x003be229: add BYTE PTR [eax],al

# Programação Genética + JIT

- Resultados
  - Antes de novas operações serem adicionadas, o código em JIT chegava a ser mais de 75x mais rápido do que o interpretado.

# Programação Genética + JIT

- Implementação do JIT
  - A implementação das árvores conta com:
    - Operadores: +, -, \*, /
    - Funções: sin, cos, max, min, abs
    - Opera sobre um número arbitrário de variáveis

# Programação Genética + JIT

- Programa desenvolvido:
  - Aproximador de funções
  - Deve-se calcular a diferença entre a função fornecida e as expressões do algoritmo genético em muitos pontos de um intervalo
    - Ideal para o benchmark das expressões compiladas em tempo real

# Programação Genética + JIT

- Implementação com o algoritmo genético
  - Cada indivíduo possui seu próprio assembler
  - Apenas ao sofrer uma mutação ou crossover o código é recompilado, para novo cálculo do fitness
  - Operação de crossover: troca ou duplicação de sub-árvores
  - Operação de mutação: folhas mutam para outras folhas, troncos para outros troncos

# Programação Genética + JIT

- Implementação com o algoritmo genético
  - Método de seleção: torneio de dois
    - Ao experimentar o "melhor cruza com todos", a carga genética do melhor acabava sendo viral e destruía com toda a variedade da população.
  - Indivíduos que geram divisões por zero ou outras aberrações matemáticas são substituídos.

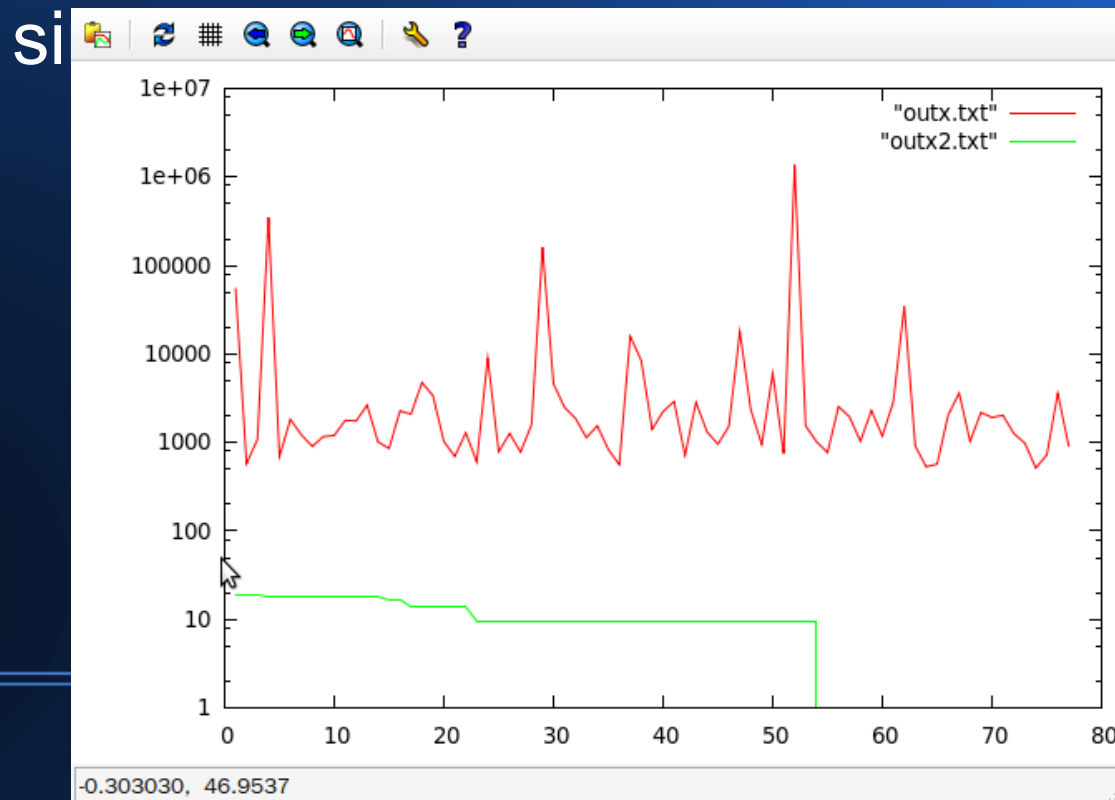
# Programação Genética + JIT

- Implementação com o algoritmo genético
  - Genocídio após 100 gerações sem melhoria no melhor fitness



# Programação Genética + JIT

- Resultados observados:
  - Rápida convergência para funções que podem ser expressas com as árvores:



# Programação Genética + JIT

- Resultados observados:
  - Convergência mais demorada para funções que não podem ser expressas pela árvore:  
 $e^{(-x^2)}$

