



Programação Genética Fortemente Tipada com Condições Sintáticas

Rafael Castro

rafaelcgs10@gmail.com

Departamento de Ciência da Computação
Centro de Ciências e Tecnológicas
Universidade do Estado de Santa Catarina

4 de Dezembro de 2017

Método básico STGP

- Programação Genética Fortemente Tipada
- Utiliza tipos para reduzir o espaço de busca: em um caso, o espaço de busca são 10^{27} possíveis, mas somente 10^{12} são programas bem tipados.
- PolyGP utiliza Cálculo Lambda Tipado (sem abstração lambda). Recursão limitada apenas para o nome da função a ser evoluída.
- O polimorfismo paramétrico (Damas-Milner) tem papel fundamental: como funções podem ser aplicadas à argumentos de vários tipos, então há um grande reuso do material genético.

Objetivo: condições sintáticas

O objetivo é verificar o impacto do uso de condições sintáticas que evitam explorar programas que não são possíveis soluções.

Por exemplo, a função `if c e1 e2` pode ter a exata mesma sub-árvore para `e1` e `e2`, ou seja, o valor de retorno do `then` é o mesmo do `else`. Portanto, pode-se considerar a condição $e1 \neq e2$.

As condições podem ser custosas computacionalmente, pois podem necessitar comparar partes de programas e aplica-las durante todo o processo evolucionário causaria um considerável aumento de tempo da evolução.

Problema map

Mapear uma função sobre uma lista:

```
map c2i ['a'..'g'] = [0..6]
```

```
T = {l::[a], []::[a], f::G1->G2}
```

```
F = {head::[a]->a, tail::[a]->[a],
      ()::a->[a]->[a], null::[a]->Bool,
      map::(G1->G2)->[G1]->[G2],
      if::Bool->a->a->a, f::G1->G2}
```

```
CS = {if c e1 e2 tal que e1 ≠ e2,
      head l tal que l ≠ [],
      tail l tal que l ≠ [] }
```

Função *fitness* do problema map

$$\begin{aligned} f(I, I') = & -2 \cdot |length(I) - length(I')| \\ & + \sum_{e \in I} 10 \cdot (2^{dist(e, I')}) \\ & - (10 + 2 \cdot length(I)) \cdot rtError \\ & - (10 + 2 \cdot length(I)) \cdot reError, \end{aligned} \tag{1}$$

onde $dist(e, I')$ é ∞ quando $e \notin I'$, ou caso contrário é a distância de e até o elemento de índice e . $rtError$ e $reError$ representam, respectivamente, erros de computações indefinidas e computações que excederam o limite de chamadas recursivas, e tem valor 1 quando acontecem e 0 caso contrário.

Fitness normalizado!!!

Parâmetros de teste do problema map

- Populações de 250, 500 e 1000 indivíduos.
- Com e sem as condições.
- Taxa de mutação de 4%.
- Taxa de *crossover* de 100%.
- Número de gerações de 200.
- Limite de chamadas recursivas de 9.
- Altura máxima das árvores de 6.
- Geração da população pelo método *grow*.
- 50 execuções de teste.

Convergência do problema map

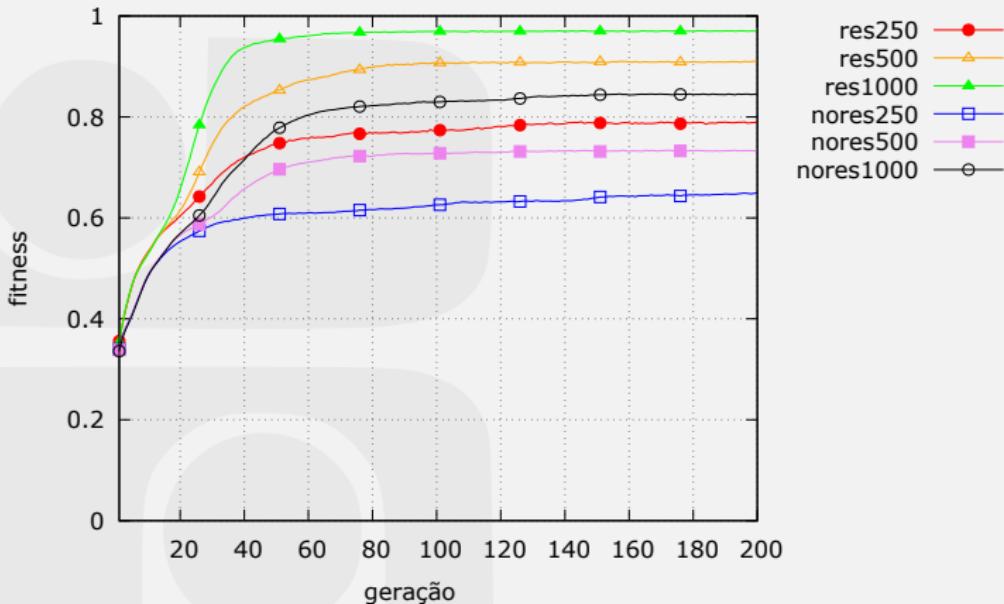


Figura: Média do fitness médio de 50 testes.

Desvio Padrão da convergência

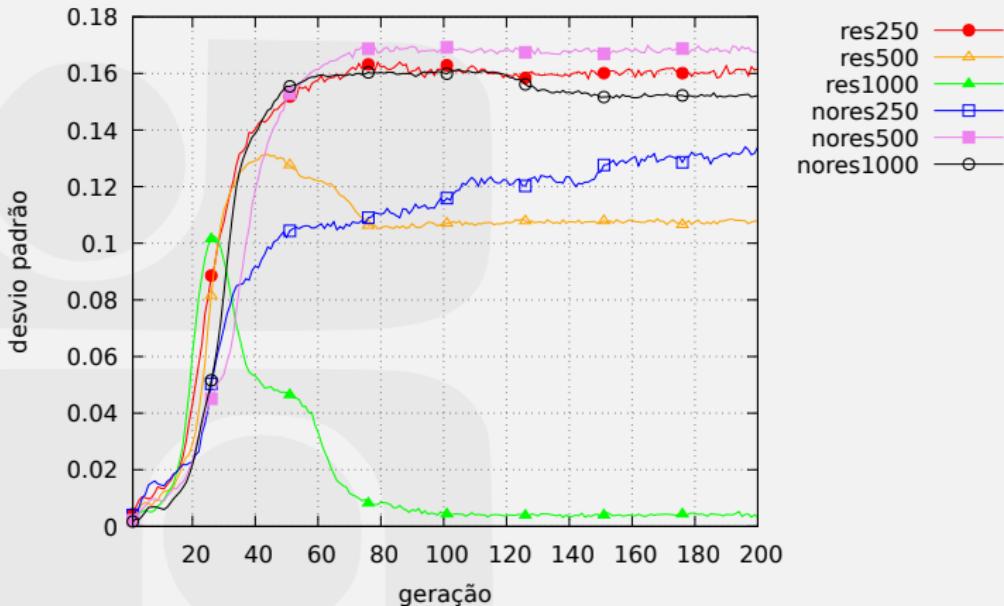


Figura: Desvio padrão do fitness médio de 50 testes.

Boxplot da convergência

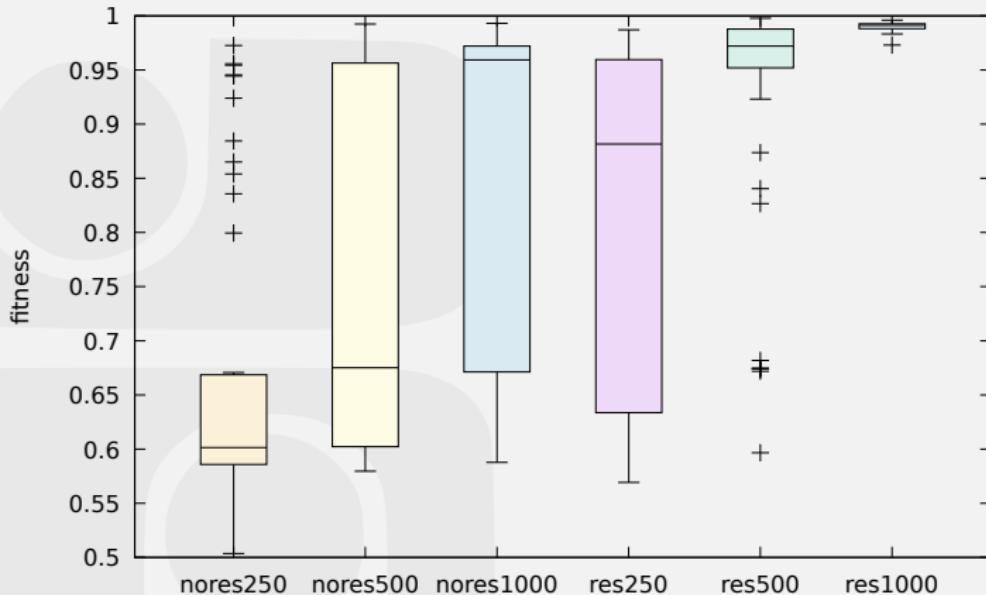


Figura: Boxplot do *fitness* da última geração.



Wilcoxon Pareado com cs1000

Wilcoxon Pareado com os valores das médias do *fitness* da última geração dos 50 testes

	nocs250	nocs500	nocs1000	cs250	cs500
w	$1.11 \cdot 10^{-9}$	$1.6 \cdot 10^{-9}$	$2.32 \cdot 10^{-9}$	$1.18 \cdot 10^{-9}$	$1.62 \cdot 10^{-7}$

Tabela: Valores do teste de Wilcoxon para média dos *fitness* da última geração.



Taxa de acerto

	nores250	nores500	nores100	res250	res500	res1000
Acerto	24%	42%	78%	78%	96%	100%

Tabela: Taxa de acerto da solução ótima.

Exemplo de resultado:

```
map f l = if null l then [] else f (head l) : map f (tail l)
```

Avaliações de *fitness* para encontrar o ótimo

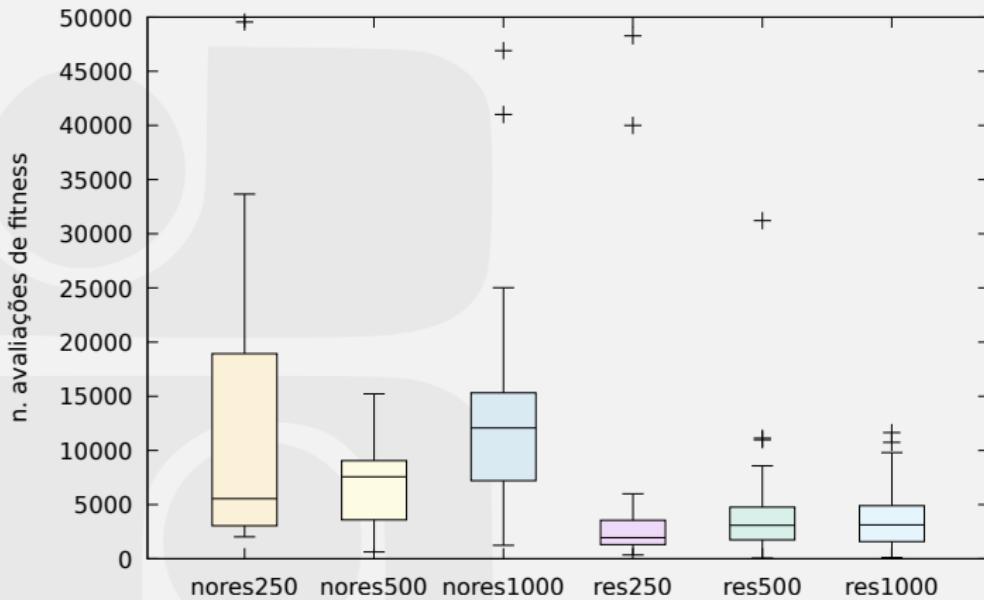


Figura: Boxplot do número de avaliações de *fitness* para encontrar a solução ótima.

Tempo médio de execução

	nores250	nores500	nores1000	res250	res500	res1000
Média	6.01s	13,74s	31.58s	8,28s	20,9s	51,38s

Tabela: Média do tempo de execução.



Considerações finais

- O método utilizado foi bem sucedido para resolver o problema map.
- O uso das condições sintáticas teve impacto positivo na convergência, desvio padrão da convergência, taxa de acerto e número de avaliações de *fitness* para encontrar o ótimo.
- Seria interessante obter informações sobre a diversidade da população.
- Estou implementando a evolução da função fibonacci como segundo problema.
- Algumas ideias: resolver problemas da maratona e evoluir funções clássicas.