# Aprendizagem Curricular Aplicada à Programação Genética Cartesiana para Otimização Lógica de Circuitos Digitais

Naiara Sachetti
Universidade Federal de Santa Catarina
Florianopolis, Santa Catarina, Brasil
naiara.sachetti@grad.ufsc.br

Bryan Martins Lima
Universidade Federal de Santa Catarina
Florianopolis, Santa Catarina, Brasil
bryan.l@grad.ufsc.br

Augusto Berndt
Universidade Federal de Santa Catarina
Florianopolis, Santa Catarina, Brasil
augusto.berndt@posgrad.ufsc.br

Cristina Meinhardt
Universidade Federal de Santa Catarina
Florianopolis, Santa Catarina, Brasil
cristina.meinhardt@ufsc.br

Jonata Tyska Carvalho
Universidade Federal de Santa Catarina
Florianopolis, Santa Catarina, Brasil
jonata.tyska@ufsc.br

## RESUMO

Atualmente, o número elevado de entradas em circuitos digitais tem se tornado um problema cada vez mais comum, demandando novas soluções para a otimização lógica dos mesmos. Uma técnica que vem sendo utilizada nos últimos anos é a de *Logic Learning*, que tem como base o uso de técnicas de *Machine Learning* (ML) para a geração de circuitos aproximados a partir de descrições parciais das funções lógicas. Uma das técnicas de ML já utilizada em *Logic Learning* é a Programação Genética Cartesiana (CGP). Apesar de fluxos de síntese lógica baseados em CGP já terem se mostrado efetivos, podem ter dificuldades em atingir uma evolucionabilidade satisfatória dentro de uma restrição de tempo de execução e para certas funções. Neste contexto, o presente trabalho busca investigar a aplicação de uma técnica denominada aprendizagem curricular (*Curriculum Learning*) para melhorar a evolucionalidade, convergindo para uma melhor acurácia. Para avaliar de forma preliminar a solução proposta foi utilizada uma porção dos *benchmarks* da competição de síntese lógica da conferência IWLS de 2020, através dos quais observou-se que, quando são priorizados exemplos de treinamento com dificuldade igualitariamente distribuída ou exemplos mais difíceis, a técnica de aprendizado curricular pode trazer benefícios para o processo evolutivo. Entre os resultados que colaboram para esta hipótese estão ganhos de até 20% na acurácia dos circuitos gerados (se considerada uma diferença absoluta) e *benchmarks* que só apresentaram ganhos em acurácia. Ainda assim, demais resultados evidenciam a importância de uma avaliação mais detalhada sobre a abordagem.

## PALAVRAS-CHAVES

Programação Genética Cartesiana, *Logic Learning*, *Machine Learning*, *Curriculum Learning*, otimização lógica

## 1 INTRODUÇÃO

Com o passar dos anos, a implementação de funções lógicas complexas em circuitos digitais tem se tornado cada vez mais comum. Tal cenário demanda novas soluções para ferramentas de otimização lógica, já que problemas que exigem tais funções, como redes neurais, normalmente possuem um número elevado de entradas. Neste contexto, uma técnica que vem sendo utilizada recentemente é a de *Logic Learning*, que se baseia na utilização de algoritmos de *Machine Learning* (ML) para a geração de circuitos aproximados, ou seja, circuitos capazes de generalizar o comportamento de funções descritas por tabelas-verdade especificadas meramente de forma parcial.

Uma das técnicas de ML já utilizadas em *Logic Learning* é a Programação Genética Cartesiana (*Cartesian Genetic Programming* - CGP), uma técnica de computação evolutiva que utiliza grafos dirigidos acíclicos representados como uma grade de duas dimensões de nodos computacionais para representar programas [6]. Tipicamente, a cada geração, uma população de programas é avaliada para que se estabeleça a acurácia de cada um de seus indivíduos e, em seguida, uma nova população é formada pelo programa com maior acurácia (ou seja, que produz o maior número de saídas esperadas) e novos indivíduos, chamados de filhos, gerados a partir de mutações de tal programa [6]. Em *Logic Learning*, tais nodos representam portas lógicas, de forma que os circuitos podem ser representados por AIGs (do inglês, AND-Inverter Graphs, Grafos de ANDs e Inversores).

Um fluxo de otimização lógica baseado em CGP é apresentado em [2] e [3], com resultados promissores para boa parte das funções lógicas consideradas, quando levados em conta simultaneamente acurácia e tamanho dos circuitos gerados. Entretanto, restam situações em que a solução não produz circuitos com uma acurácia satisfatória. Sendo assim, fluxos de otimização lógica baseados em CGP poderiam se beneficiar da aplicação de novos mecanismos potencialmente capazes de melhorar sua evolucionabilidade, ou seja, a sua capacidade de continuar encontrando soluções melhores. Seguindo esta direção, decidiu-se investigar o uso da técnica de *Curriculum Learning* (CL) [5].

Como formalizado em [1], um *curriculum* pode ser visto, de forma abstrata, como uma sequência de critérios de treinamento, com cada critério sendo associado com um conjunto diferente de pesos nos exemplos de treinamento. Os autores de [4] trazem este conceito para o contexto de treinamento evolucionário de agentes corporificados (isto é, que possuem um corpo e estão situados em um ambiente), usando CL para manipular as condições ambientais que desafiam as fraquezas dos agentes em evolução, mantendo um nível apropriado de dificuldade.

Neste trabalho, aplicou-se uma estratégia de *Curriculum Learning* semelhante à utilizada em [4] a um fluxo de otimização lógica baseado em CGP inspirado em [2] e [3]. No presente resumo são apresentados resultados de experimentos preliminares que visam entender se tal abordagem pode ser efetiva para melhorar a acurácia dos circuitos em evolução.

## 2 SOLUÇÃO PROPOSTA

A Figura 1 mostra o fluxo de otimização lógica baseado em CGP utilizado como base para os experimentos realizados. Tal fluxo parte de populações iniciais aleatórias, seleciona os circuitos com maior acurácia e avalia a acurácia final do circuito gerado. Durante a etapa de seleção, a avaliação dos circuitos é feita a partir de *mini-batches*, compostos por uma parcela aleatoriamente selecionada das linhas da tabela-verdade utilizada como conjunto de treinamento. A quantidade de linhas que compõem cada *mini-batch* é dada pelo hiperparâmetro *batch size*, enquanto a quantidade de gerações que um mini-batch deve permanecer inalterado é definida pelo hiperparâmetro *change each*.

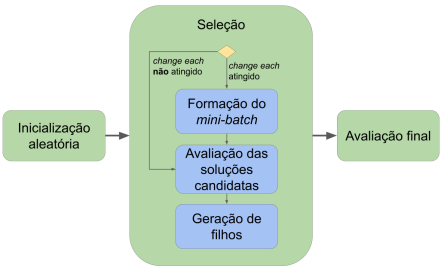



**Figura 1: Fluxo de otimização baseado em CGP utilizado como base.**

A estratégia de *Curriculum Learning* (CL) proposta neste trabalho substitui a seleção aleatória de linhas para os *mini-batches*. De modo semelhante à abordagem utilizada em [4], tal seleção é feita com base em uma *função de dificuldade* e em um valor de dificuldade atribuído para cada uma das linhas da tabela-verdade. As alterações no fluxo necessárias para o suporte de CL foram feitas diretamente no código original, na linguagem C++.

Na aplicação de *Curriculum Learning*, o valor de dificuldade é definido por meio da Equação 1, de modo que $H_i$ é o número de vezes que uma linha foi corretamente solucionada por circuitos em avaliação nas gerações anteriores, enquanto $E_i$ é o número total de vezes que uma linha participou de avaliações de circuitos.

$$D_i = \frac{H_i}{E_i} \tag{1}$$

Já a função de dificuldade define os intervalos de dificuldade nos quais as linhas que compõem os *mini-batches* devem se encaixar. O limite de dificuldade para cada posição do *mini-batch* é dado pela Equação 2, sendo $b$ o índice da posição do *mini-batch* e $m$ o valor de *batch size*.

O valor inicial de dificuldade para cada uma das linhas é definido através de uma avaliação das mesmas antes da primeira geração de seleção de circuitos. Esta avaliação é feita com a população inicial de circuitos gerada aleatoriamente e que deverá, logo em seguida, passar pela seleção.

$$L_b = \frac{f(b)}{f(m)} \tag{2}$$

## 3 AVALIAÇÃO E RESULTADOS

Para avaliar preliminarmente a abordagem idealizada utilizou-se dezesseis *benchmarks* da competição de síntese lógica da conferência IWLS de 2020. O critério de escolha foi o tamanho das entradas, considerando os recursos computacionais disponíveis no momento. Assim, considerou-se apenas os *benchmarks* com no máximo 20 entradas.

As funções de dificuldade consideradas foram linear e quadrática. No primeiro caso, são gerados mini-batches com uma dificuldade igualitariamente distribuída, evitando que somente linhas fáceis formem o conjunto. No segundo caso, prioriza-se a seleção de linhas difíceis. Em ambos os casos, garante-se que os circuitos, durante o processo de seleção, sejam expostos a linhas não triviais e possam se adaptar às mesmas. Assim, tal técnica evita que os indivíduos sejam mal avaliados e haja *overfitting*.

Já os hiperparâmetros considerados são listados na Tabela 1, onde a *População* indica a quantidade de circuitos gerados aleatoriamente antes da seleção e utilizados durante a mesma, a *Profundidade lógica* define o número máximo de nodos que podem ser encadeados nos circuitos gerados, enquanto os demais hiperparâmetros indicam a quantidade de gerações utilizada na seleção de indivíduos, o tamanho dos *mini-batches* e a frequência com que estes devem ser substituídos.

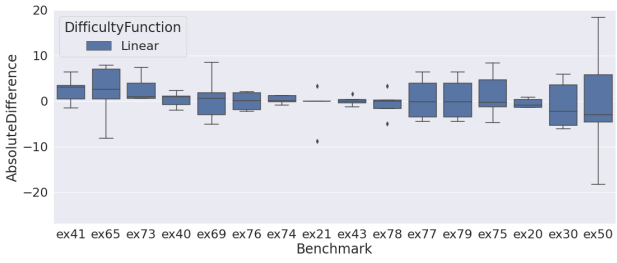
Ainda, a fim de minimizar a influência da estocasticidade nos resultados obtidos, para cada um dos *benchmarks* foram rodadas cinco evoluções com sementes diferentes para a geração de números aleatórios durante o processo.

**Tabela 1: Hiperparâmetros utilizados.**

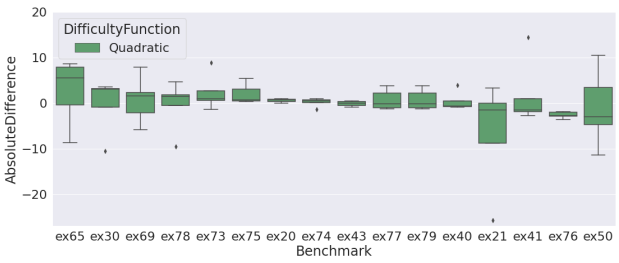
| Hiperparâmetro | Valor |
|---|---|
| População | 5 |
| Profundidade lógica | 250 |
| Gerações | 50.000 |
| *Batch size* | 64 |
| *Change each* | 250 |

A Figura 2 apresenta as diferenças absolutas entre acurácias obtidas a partir das versões do fluxo com e sem CL, separadas por *benchmark*. Na Figura 2a, que considera uma função de dificuldade linear, o ex73 apresentou somente ganhos em acurácia e, como é possível observar através do ex50, ganhos de até aproximadamente 20% foram observados. Ainda assim, também foram observadas perdas de aproximadamente 20%.

Já a Figura 2b apresenta as diferenças absolutas tendo em conta uma função de dificuldade quadrática. Aqui, pode-se destacar o ex75, que somente teve ganhos, além dos limites de ganho e perda, de aproximadamente 15% e 25%, respectivamente.




**(a) Para uma função de dificuldade linear.**




**(b) Para uma função de dificuldade quadrática.**

**Figura 2: Diferenças absolutas com relação às acurácias em CGP-noCL.**

## 4 CONSIDERAÇÕES FINAIS

Neste trabalho, apresentou-se resultados preliminares da aplicação de *Curriculum Learning* a um fluxo de otimização lógica baseado em CGP. A partir dos dados observados, pode-se concluir que a utilização de *Curriculum Learning* em conjunto com CGP tem a possibilidade de ser uma estratégia eficaz de aumento da acurácia dos circuitos gerados. Sendo assim, uma avaliação mais detalhada sobre a abordagem apresentada é não somente necessária, como também valorosa.

Para tanto pretende-se otimizar os hiperparâmetros utilizados, verificando quais valores produzem os melhores resultados, bem como avaliar quais características dos próprios *benchmarks* podem definir se a técnica proposta trará ou não maior efetividade. Pretende-se ainda explorar novas funções de dificuldade como a função raiz, por exemplo.

## REFERÊNCIAS

[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum Learning. In *Proceedings of the 26th Annual International Conference on Machine Learning* (Montreal, Quebec, Canada) *(ICML '09)*. Association for Computing Machinery, New York, NY, USA, 41–48. https://doi.org/10.1145/1553374.1553380

[2] Augusto Berndt, Brunno A. de Abreu, Isac S. Campos, Bryan Lima, Mateus Grellert, Jonata T. Carvalho, and Cristina Meinhardt. 2021. Accuracy and Size Trade-off of a Cartesian Genetic Programming Flow for Logic Optimization. In *Proceedings of the 34th Symposium on Integrated Circuits and Systems Design* (Brazil) *(SBCCI '21)*. https://doi.org/10.1109/SBCCI53441.2021.9529968

[3] Augusto Berndt, Brunno A. de Abreu, Isac S. Campos, Bryan Lima, Mateus Grellert, Jonata T. Carvalho, and Cristina Meinhardt. 2022. A CGP-based Logic Flow: Optimizing Accuracy and Size. *Journal of Integrated Circuits and Systems (JICS)* (2022). https://doi.org/10.29292/jics.v17i1.546

[4] Nicola Milano and Stefano Nolfi. 2021. Automated curriculum learning for embodied agents a neuroevolutionary approach. *Scientific Reports* 11, 8985 (April 2021), 1–14. https://doi.org/10.1038/s41598-021-88464-5

[5] Nicola Milano, Paolo Pagliuca, and Stefano Nolfi. 2019. Robustness, evolvability and phenotypic complexity: insights from evolving digital circuits. *Evolutionary Intelligence* 12, 1 (2019), 83–95.

[6] Julian F. Miller. 2011. *Cartesian Genetic Programming*. Springer Berlin Heidelberg, Berlin, Heidelberg, 17–34. https://doi.org/10.1007/978-3-642-17310-3_2